Björn Hackel

Bruder Klaus Str. 25
78467 Konstanz

Technikerarbeit 2007/08
Dokumentation V1.0

GPS-Tracker

Betreuende Lehrer: Herr Sperling, Herr Groß

Inhaltsverzeichnis

1. **Projektbeschreibung 4**

2. **Funktionsübersicht 6**
2.1 Handgerät 7
2.2 PC Software 9

3. **GPS 11**
3.1 Allgemeines 12
3.2 Positionsbestimmung 12
3.3 Satellitengeometrie und Genauigkeit 13
3.4 Aufbau der Satellitenverbindung 14

4. **Projektentstehung 16**
4.1 Zeitlicher Ablauf 18
4.2 Hardwareentwicklung 21
4.3 Softwareentwicklung (µC) 24
4.4 Softwareentwicklung (PC) 28

5. **Technische Beschreibung 31**
5.1 Grundlagen 32
5.1.1 NMEA 0183 Format 32
5.1.2 Keyhole Markup Language 34
5.2 Hardware 35
5.2.1 Blockschaltbild 35
5.2.2 Schaltplan 36
5.2.3 Schaltplan Erläuterungen 38
5.2.4 Technische Daten 41
5.2.5 Platinenlayout 41
5.2.6 Stückliste 43
5.3 Software (µC) 44
5.3.1 Allgemeines 44
5.3.2 Der grobe Ablauf 47
5.3.3 main.c 47
5.3.4 einlesen.c 52
5.3.5 ausgabe.c 58
5.3.6 tracking.c 62
5.3.7 zeitumstellung.c 65
5.3.8 eeprom.c 68
5.3.9 funktionen.c 68
5.4 Software (PC) 70
5.4.1 Allgemeines 70
5.4.2 Klassenübersicht 71
5.4.3 Form1_Main.vb 74

5.4.3.1 ToolStripMenüItem_Click 74
5.4.3.2 Button_Click 77
5.4.3.3 Load and Close 78
5.4.3.4 SelectedItemChanged 79
5.4.3.5 Timer_Tick 79
5.4.3.6 DataReceived 79
5.4.4 Form2_Info.vb 80
5.4.5 Form2_Einstellungen.vb 81

6. Bedienungsanleitung 82
6.1 Handgerät 83
6.1.1 Legende 83
6.1.2 Menüaufbau 84
6.1.3 Bedienung 85
6.1.4 Akku laden 86
6.2 PC Software 87
6.2.1 Erstinbetriebnahme 87
6.2.2 Bedienung 89
6.3 Kontrollelemente 90
6.4 Mögliche Fehler 90

7. Fazit 91

8. Anhang 93
8.1 Eidesstattliche Erklärung 94
8.2 Glossar 95
8.3 Quellen 97
8.4 Inhalt der beigefügten CD 97

Kapitel 1
Projektbeschreibung

=> Was ist der GPS-Tracker?

Mit dem „GPS-Tracker“ wurde ein Projekt entwickelt, welches imstande ist eine Strecke aufzuzeichnen und diese in „Google Earth“ grafisch darzustellen. Darüber hinaus können über das Handgerät bei Satellitenempfang und während der Aufnahme, weitere Informationen abgerufen werden. Diese werden im nächsten Kapitel genauer beschrieben.

Das Projekt besteht aus zwei Teilen:

1. Das Handgerät nimmt Wegpunkte auf und zeigt unterschiedliche Informationen über die aktuelle Position und die zurückgelegte Strecke an. Die Wegpunkte können über eine USB Schnittstelle zu einem PC übertragen werden.

2. Die PC Software nimmt die übertragenen Wegpunkte entgegen und konvertiert sie in ein Format, welches in „Google Earth“ eingebunden werden kann.

Bild 1 ist ein Beispiel für eine solche Streckenaufzeichnung.

Bild 1

Für jeden Wegpunkt wird hier ein grünes, doppeltes Dreieck, mit fortlaufender Nummerierung abgebildet.

Kapitel 2
Funktionsübersicht

- **Handgerät**
- **PC Software**

=> Was kann der GPS-Tracker?

2.1 Handgerät

Bild 2

In der nachstehenden Auflistung werden die Funktionen des Handgerätes aufgeführt.

Im Hauptmenü sind folgende Menüpunkte zu finden:

- Die Position, bestehend aus Längen- und Breitengrad
- Die Höhe über dem Meeresspiegel
- Die aktuelle Geschwindigkeit
- Die Anzahl der empfangenen Satelliten
- Das Datum und die Uhrzeit (deutsche Zeitzone)
- Eine Übersicht, bestehend aus Position, Höhe und Geschwindigkeit
- Die Verknüpfung zum Trackingmenü

Im Trackingmenü werden folgende Menüpunkte angezeigt:

- Verknüpfung zum Aufnahmemenü
- Die aufgezeichneten Strecken, bestehend aus der horizontalen-, der vertikalen und der realen Strecke
- Die aufgezeichneten Geschwindigkeiten, bestehend aus der Durchschnitts-, der Maximal- und der durchschnittlichen vertikalen Geschwindigkeit
- Die aufgezeichneten Zeiten, bestehend aus der Aufnahmedauer, der Startzeit und der Endzeit des Tracks
- Die Startposition, bestehend aus der Höhe und dem Längen- und Breitengrad
- Die Option der Datenübertragung zum PC
- Die Option zur Rückkehr ins Hauptmenü

Die Werte des letzten Tracks bleiben bis zum Start des nächsten Tracks erhalten und sind im Menü abrufbar. Mit dem Start einer neuen Aufzeichnung werden die alten Daten überschrieben.

Im Aufnahmemenü kann zwischen folgenden Optionen gewählt werden:

- Neue Aufzeichnung starten
- Angefangene Aufzeichnung fortsetzen
- Ins Trackingmenü zurückkehren

Im Statusmenü werden folgende Werte angezeigt:

- Die Genauigkeit der allgemeinen Positionsbestimmung
- Die Genauigkeit der horizontalen Positionsbestimmung
- Die Genauigkeit der Höhenbestimmung
- Der Batteriestatus
- Die Empfangsstärke
- Die Anzahl der bereits gespeicherten Wegpunkte (falls nicht aufgezeichnet wird, wird der Wert der letzten Aufzeichnung angezeigt)

Das Gerät wird über drei Taster gesteuert. Mit einem der Taster kann der Menüpunkt ausgewählt werden. Mit einem anderen Taster kann dieser Menüpunkt bestätigt werden, wodurch die gewählte Aktion durchgeführt wird. Hält man den dritten Taster gedrückt, erscheint die Statusanzeige auf dem Display.

Zur besseren Bedienbarkeit leuchten 4 LEDs in folgenden Situationen:

- eine Aufzeichnung läuft
- kein Satellitenempfang
- schwacher Batteriestatus
- USB Verbindung zu einem Rechner

2.2 PC-Software

Bild 2 zeigt die Benutzeroberfläche der PC Software.

Bild 3

Für die Anzeige der Koordinaten in „Google Earth“ müssen die gespeicherten Wegpunkte zum Rechner übertragen werden. Die GPS-Tracker Software auf dem PC konvertiert die Daten in das „Google Earth“ Format.

In der folgenden Auflistung werden die Funktionen der Software gezeigt:

Hauptfunktionen:

- Konvertierung der Wegpunkte des Handgeräts in eine KML-Datei
- Anzeige, ob das Gerät verbunden ist
- Anzeige der bereits übertragenen Tracks
- Beim Markieren einer Datei aus der Liste, werden das Übertragungsdatum, die Anzahl der Wegpunkte und die Länge des Tracks angezeigt
- Durch einen Doppelklick auf die markierte Datei, wird die Fehleranzahl der Übertragung angezeigt
- Die markierte Datei kann über einen „Button“ geöffnet und über einen anderen gelöscht werden
- Beim ersten Start der Anwendung, wird nach einem Speicherort für die Tracks gefragt. Dieser muss dann eingerichtet werden

Optionen in der Menüleiste:

- Über „Datei – Beenden“ kann das Programm beendet werden
- Über „Bearbeiten – Aktualisieren“, wird die Liste mit den Tracks aktualisiert und alphabetisch sortiert

- Über „Bearbeiten – Gerät suchen“, kann das Gerät gesucht und verbunden werden, wenn es erst nach dem Programmstart eingesteckt, oder die Schnittstelle neu konfiguriert wurde
- Über „Bearbeiten – Einstellungen“, kann die momentan eingestellte Schnittstelle und der Speicherpfad für die Tracks abgelesen und geändert werden (Die Daten werden abgespeichert und sind beim nächsten Programmstart wieder aktiv)
- Über „Info“ wird ein kurzer Hinweis zur Erstellung des Projekts angezeigt

Kapitel 3
GPS

- **Allgemeines**
- **Positionsbestimmung**
- **Satellitengeometrie und Genauigkeit**
- **Einschaltzeiten**

=> Was ist GPS und wie funktioniert es?

3.1 Allgemeines

Die folgende Beschreibung wurde der Website www.kowoma.de entnommen und geringfügig abgeändert.
Das GPS-System ist ein vom amerikanischen Verteidigungsministerium (DOD; Department of Defense) ersonnenes, realisiertes und betriebenes System, das aus 30 aktiven Satelliten besteht, welche die Erde in einer nominellen Höhe von 20.200 km umkreisen. GPS Satelliten senden Signale aus, welche die genaue Ortsbestimmung eines GPS Empfängers ermöglichen. Die Empfänger können ihre Position ermitteln, wenn sie feststehend sind, sich auf der Erdoberfläche in der Erdatmosphäre oder in niederen Umlaufbahnen bewegen.
GPS wird sowohl in der Luft-, Land- und Seefahrtnavigation als auch bei der Landvermessung und anderen Anwendungen eingesetzt, bei denen es auf genaue Positionsbestimmung ankommt. Das GPS-Signal wird jedem kostenlos zur Verfügung gestellt.
Der eigentlich Name des Systems ist NAVSTAR (Navigation System for Timing and Ranging), bekannt ist es aber nur als GPS (Global Positioning System).

3.2 Positionsbestimmung

Stark vereinfacht gesagt sendet jeder Satellit eine Nachricht der Art: "Ich bin Satellit Nr. X, meine Position ist gerade Y und diese Nachricht wurde zum Zeitpunkt Z versandt“. Zusätzlich sendet der Satellit noch Informationen über seine Umlaufbahn (und die der anderen Satelliten). Diese Bahndaten (Ephemeriden- und Almanachdaten) werden vom GPS-Empfänger gespeichert und für spätere Rechnungen verwendet.

Um die Position zu bestimmen, vergleicht der GPS-Empfänger die Zeit, zu der das Signal ausgesandt wurde mit der Zeit, zu der das Signal empfangen wurde. Aus dieser Zeitdifferenz kann die Entfernung des Satelliten berechnet werden. Werden von weiteren Satelliten Messungen hinzugefügt, so kann die aktuelle Position durch Trilateration (Entfernungsmessung von drei Punkten aus) bestimmt werden. Mit wenigstens drei Satelliten kann der GPS Empfänger seine Position auf der Erdoberfläche berechnen. Dies wird "**2D position fix**" (zweidimensionale Positionsbestimmung) genannt. Zweidimensional deshalb, weil der Empfänger sich direkt auf der Erdoberfläche (also einer rechnerisch zweidimensionalen Fläche) befindet. Mit Hilfe von vier oder mehr Satelliten kann ein "**3D position fix**", also die absolute Position im Raum oder eben zusätzlich die Höhe über der Erdoberfläche bestimmt werden.

Durch ständige Neuberechnung der aktuellen Position kann der GPS Empfänger auch genau die Geschwindigkeit und die Bewegungsrichtung (als "ground speed" und "ground track" bezeichnet) berechnen.
Eine andere Möglichkeit der Geschwindigkeitsmessung ist das Ausnutzen des

Dopplereffekts, der durch die Bewegung der übermittelten Signalen auftritt. Dies funktioniert nach dem gleichen Prinzip, wie ein Beobachter ein Martinshorn als höheren Ton wahrnimmt, wenn es sich auf ihn zu bewegt und als tieferen Ton, wenn es sich von ihm weg bewegt.

3.3 Satellitengeometrie und Genauigkeit

Ein Faktor, der die Genauigkeit der Positionsbestimmung beeinflusst ist die "Satellitengeometrie". Einfach gesagt bezieht sich "Satellitengeometrie" auf die vom Empfänger aus gesehene Stellung der gerade empfangenen Satelliten zueinander im Raum.

Wenn ein Empfänger beispielsweise vier Satelliten empfängt und alle vier Satelliten sind nur im Nordwesten, so ergibt sich daraus eine "schlechte Geometrie". Unter Umständen kommt überhaupt keine Positionsbestimmung zustande. Wenn alle Entfernungsmessungen aus der gleichen Richtung erfolgen, kann keine Position bestimmt werden. Selbst wenn der Empfänger eine Positionsbestimmung durchführen kann, so kann der Fehler im Bereich von 100 bis 150 Metern liegen.
Sind hingegen die vier empfangenen Satelliten gleichmäßig über den gesamten Himmel verteilt, so wird die Positionsbestimmung wesentlich genauer. Angenommen die Satelliten befinden sich jeweils im Norden, Osten, Süden und Westen, sind also in 90° Abständen angeordnet, so ist die "Satellitengeometrie" sehr gut.

Es gibt ein Maß für die Genauigkeit der Messwerte, die meist ein Kombinationswert verschiedener Faktoren sind. Für die "Güte" der Satellitengeometrie sind die DOP-Werte (dilution of precision; Verschlechterung der Genauigkeit) sehr verbreitet.
Je nachdem, welche Daten bei der Berechnung herangezogen werden unterscheidet man zwischen verschiedenen DOP-Werte:

- GDOP (Geometric Dilution Of Precision); Gesamtgenauigkeit; 3D-Koordinaten und Zeit
- **PDOP** (Positional Dilution Of Precision); Positionsgenauigkeit; 3D-Koordinaten
- **HDOP** (Horizontal Dilution Of Precision); Horizontalgenauigkeit; 2D-Koordinaten
- **VDOP** (Vertical Dilution Of Precision); Vertikalgenauigkeit; Höhe
- TDOP (Time Dilution Of Precision); Zeitgenauigkeit; Zeit

So sind HDOP-Werte unter 4 sehr gut, über 8 jedoch schlecht. Die HDOP Werte werden schlechter, wenn sich die empfangenen Satelliten hoch am Himmel befinden. VDOP Werte hingegen sind nicht so gut, wenn sich die Satelliten sehr nahe am Horizont befinden. Die PDOP Werte sind am besten, wenn sich ein Satellit über Kopf und drei weitere gleichmäßig am Horizont verteilt befinden. Für eine gute Bestimmung sollte der GDOP-Wert nicht über 5 sein. Die PDOP, HDOP und VDOP Werte werden im NMEA-Datensatz $GPGSA ausgegeben.

Zusätzlich wird die Positionsbestimmung noch durch weitere Fehlerquellen beeinflusst. Die Fehlerbilanz sieht etwa folgendermaßen aus, wobei die Werte keine festen Größen, sondern durchaus Schwankungen unterworfen sind. Die angegebenen Werte sind circa-Werte.

- Störungen durch die Ionosphäre ±5 Meter
- Schwankungen der Satellitenumlaufbahnen ±2.5 Meter
- Uhrenfehler der Satelliten ±2 Meter
- Mehrwegeeffekt ±1 Meter
- Störungen durch die Troposphäre ±0.5 Meter
- Rechnungs- und Rundungsfehler ±1 Meter

Insgesamt ergibt sich daraus ein Fehler von ± 12 Metern. Mit Korrektur durch Systeme wie WAAS und EGNOS, wodurch vor allem Ionosphäreneffekte aber auch Umlaufbahnen und Uhrenfehler reduziert werden, wird der Fehler auf etwa ± 3 - 5 Meter reduziert.

3.4 Aufbau der Satellitenverbindung

Die Speicherung der Ephemeriden- und Almanach-Daten im GPS-Empfänger führt dazu, dass es unterschiedlich lange dauern kann, bis die erste Positionsbestimmung verfügbar ist. Das hängt davon ab, wie lange der GPS-Tracker keinen Empfang hatte,

War der Empfang der Signale lediglich kurz unterbrochen (z.B. Tunnelfahrt, Wald) so spricht man von Wiedererfassung (engl. reaquisition). Dies dauert nur sehr wenige Sekunden.

Von einem **Heisstart** (Hot Start) spricht man, wenn Position und Uhrzeit bekannt sind, die Almanach-Daten und die Ephemeriden-Daten aktuell sind. Dieser Fall tritt ein, wenn das Gerät innerhalb der letzten 2 - 6 Stunden am etwa gleichen Ort eine Positionsbestimmung durchgeführt hat. Dabei dauert es etwa 15 Sekunden, bis eine Positionsbestimmung (engl. position fix) verfügbar ist.

Wenn die letzte Position bekannt ist, das Almanach vorhanden und die Uhrzeit im Empfänger stimmt aber die Ephemeriden veraltet sind, spricht man von einem **Warmstart** (Warm Start). Dabei müssen nur die Ephemeridendaten aktualisiert werden und es dauert etwa 45 Sekunden bis einen Positionsbestimmung verfügbar ist. Die Ephemeriden sind veraltet, wenn mehr als etwa 2 - 6 Stunden seit dem letzten Empfang von Daten zu den momentan sichtbaren Satelliten vergangen sind.

Je mehr andere Satelliten seit dem letzten Einschalten am Himmel stehen desto länger dauert der Warmstart.

Sind weder Ephemeriden, noch Almanach-Daten noch die letzte Position bekannt, spricht man von einem **Kaltstart** (Cold Start). Es muss zunächst der Empfang aller Almanach-Daten der Satelliten abgewartet werden. Das kann bis zu 12,5 Minuten dauern.
Das gleiche Verzögerung kann auftreten, wenn der GPS-Tracker längere Zeit (Wochen) ausgeschaltet war oder mehr als etwa 300 km seit dem letzten Positionsfix bewegt wurde.

Kapitel 4
Projektentstehung

- **Zeitlicher Ablauf**
- **Hardwareentwicklung**
- **Softwareentwicklung (µC)**
- **Softwareentwicklung (PC)**

=> Wie ist der GPS-Tracker entstanden?

Zu Beginn der Technikerausbildung wurde darauf hingewiesen, dass jeder Schüler als Abschlussprojekt eine Technikerarbeit erstellen muss. Die einzige Vorgabe war, dass es ein Mikrocontroller gesteuertes Gerät sein soll.

Da mich die kabellose Übertragung von Signalen schon immer fasziniert hatte, kam mir als erstes die Idee in den Sinn, ein Funkgerät zu bauen. Dazu fand ich auch schnell mehrere Schaltpläne im Internet. Aufgrund der Tatsache, dass ich einen Amateurfunkschein benötigt hätte um dieses Projekt zu realisieren, entschloss ich mich dann aber doch dagegen.
Also suchte ich im Internet nach geeigneten Projekten. Nach intensiver Suche kam ich auf die Seite http://thomaspfeifer.net/ von, wem auch anders, Thomas Pfeifer. Dort wurde ich unter seinen Projekten auf den GPS Tracker aufmerksam.

Die Aufzeichnung einer Strecke, welche dann in „Google Earth“ dargestellt werden kann, hatte mich so fasziniert, dass ich mich sofort dazu entschied, dieses Projekt selbst zu entwickeln.
Die Umwandlung in das „Google Earth“ Format machte mir anfangs noch Sorgen. Doch schon bald darauf fand ich fertige Software zum konvertieren und das Problem war gelöst. Als mir Herr Sperling schließlich auch noch das OK gab, konnte der Spaß beginnen…

4.1 Zeitlicher Ablauf

2007

März Der erste Schritt war die Informationsbeschaffung. Es musste geklärt werden, was es für GPS Module gibt und wie diese zu handhaben sind.
Erwerb des GPS Moduls EM 411.

April - Juni Beschaffung von Informationen über die Technik der GPS Positionsbestimmung.
Recherche über die Einbindung von Tracks in „Google Earth“.
Kleineren Übungen zur Filterung von Informationen aus einem seriellen Datenstrom.

Juli Einarbeitung in die Programmiersprache „Visual Basic 2005“.
Erstellung eines Programms zur Simulation des GPS Moduls.
Versuchsaufbau mit einem Steckbrett und einem LPC-Experimentierboard.
Anschluss des GPS Moduls an den Mikrocontroller.

August Filtern der Daten des GPS Moduls.
Abgabe des Pflichtenheftes.

September Erstes Konzept für das Abrufen der Informationen aus den GPS Datensätzen über eine Menüsteuerung.
Start der Softwareentwicklung für den PC.

Oktober Anbindung des Experimentierboards an den PC.
Erste Versuche mit der Streckenberechnung zwischen zwei Koordinaten.

November Erstellung der Algorithmen für die Berechnungen der Strecke, der Aufnahmezeit, der Durchschnitts- und Maximalgeschwindigkeit.
Speicherung der Wegpunkte in dem externen EEPROM.

Dezember Erste Erstellung einer KML-Datei mit dem Programm „GPS-Track-Analyse“.
Entwicklung der Zeitumrechnung von der UTC Zeit auf die deutsche Zeitzone.

2008

Januar Entwicklung der asynchronen, seriellen Datenübertragung.
Start der Hardwareentwicklung. Erstellung des Schaltplans und des Platinenlayouts.
Erster Versuchsaufbau auf dem Steckbrett ohne Experimentierboard.

Februar Entwicklung der Spannungsversorgung.
Interpretation der KML-Sprache und erste Versuche der Erstellung einer KML-Datei mit eigener Software.
Versuchsaufbau mit geätzter Platine.
Erste Datenübertragung via USB.

März Aufnahme eines Tracks im Freien.
Feinschliff der Software.

April Erstellung des Gehäuses.
Abgabe des GPS Trackers.

Zeitaufwand

	2007						2008			
Aufgabe	Juli	August	September	Oktober	November	Dezember	Januar	Februar	März	April
Informationsphase										
Softwareentwicklung										
Hardwareentwicklung										
Hardwareerstellung										
Fehlerbehebung										

Tabelle 1

Die Fertigstellung des Projekts beanspruchte etwa 450 Stunden.

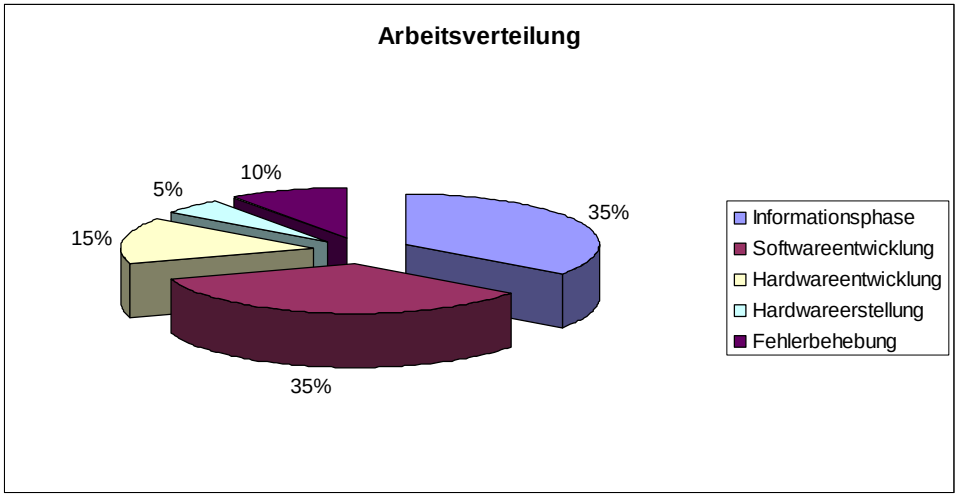

Arbeitsverteilung
10%
5%
15%
35%
35%
Informationsphase
Softwareentwicklung
Hardwareentwicklung
Hardwareerstellung
Fehlerbehebung

Bild 4

4.2 Hardwareentwicklung

GPS-Modul:

Herzstück des Gerätes ist das GPS Modul. Dieses sollte einen seriellen Datenstrom liefern und möglichst genau sein. Nach einer ausführlichen Recherche im Internet bestellte ich das GPS Modul EM-411 der Firma Navilock.
Das Modul war optimal für meine Anwendung geeignet. Es hat eine integrierte Antenne, recht kleine Abmessungen und liefert die GPS Daten als seriellen ASCII-Datenstrom. Aufgrund des empfindlichen „Sirf 3“ Chipsatzes ist es zudem sehr genau.
Ich fing also an mit diesem Modul zu experimentieren.

Spannungsversorgung:

Oberstes Kriterium für die Hardware war die Portabilität und somit auch die Größe. Es sollte also ein möglichst kleines, batteriebetriebenes Gerät werden.
Von Anfang an war klar, dass ich zwei Spannungen brauchen würde: Für den Mikrocontroller 3,3V und für das LCD sowie das GPS Modul, 5V. Eine Möglichkeit, die mir gar nicht gefiel, war die Realisierung über 4 Mignon oder Micro Batterien. Diese hätten, in meinen Augen, zuviel Platz benötigt.
Da in Handys und anderen mobilen Geräten auch Displays und andere Bauteile mit 5V Spannungsversorgung eingebaut werden, musste es noch bessere Möglichkeiten geben. So kam mir die Idee, einen „Boost Konverter“ zu verwenden.
Da ich mit solchen Spannungsversorgungen noch überhaupt keine Erfahrungen gesammelt hatte, beschloss ich die Sache erst einmal ruhen zu lassen und mich am Ende, wenn der Rest der Schaltung funktioniert, weiter damit zu beschäftigen.
Falls gegen Ende der Abgabefrist keine Zeit mehr für die Realisierung der Spannungsversorgung mit dem „Boost Konverter“ vorhanden sein sollte, hätte diese auch mit einer größeren Batterie und Festspannungsreglern aufgebaut werden können.

Mikrocontroller:

Mir war klar, dass ich für dieses Projekt viel Speicherplatz benötigen würde. Deshalb entschied ich mich für den Mikrocontroller LPC936. Er ist der größte Controller seiner Serie und im Unterricht haben wir diesen ausführlich besprochen.
Erst war geplant die SMD Variante des Mikrocontrollers zu verwenden, doch aufgrund der Komplexität der Software entschied ich mich dann für die PLCC Version. Diese hat den eindeutigen Vorteil, dass man den Controller immer wieder neu programmieren kann, ohne ihn aus der Schaltung auslöten zu müssen.
Während der Entwicklungsphase und zu Testzwecken ist dies eindeutig die bessere Lösung. Das Gerät kann nach der Fertigstellung der Software immer noch auf SMD umgerüstet werden.

Speicher:

Es sollten natürlich so viele Datensätze wie möglich abgespeichert werden. Der erste Gedanke war eine SD Karte. Diese Lösung erschien mir als optimal. Ich recherchierte eine ganze Weile im Internet, las Datenblätter und Beispiel Codes. Erst sah ich es als Herausforderung, doch als die Zeit verstrich und ich dann auch noch bemerkte das SD Karten relativ hohe Schreibeströme benötigen (bis zu 90mA), hakte ich das Thema ab und informierte mich über andere Möglichkeiten.
Speichermedien wie CF Karten oder andere, mit paralleler Ansteuerung schloss ich gleich, wegen des erhöhten Platzbedarfs aus. Es sollte also ein Speichermedium mit serieller Ansteuerung sein. Da wir zu dieser Zeit im Unterricht gerade den I²C-Bus durchnahmen, entschied ich mich bald darauf, ein serielles EEPROM zu verwenden. Diese haben zwar eine vergleichsweise sehr kleine Speicherkapazität, aber sie sind einfach anzusteuern, haben einen geringen Platzbedarf und benötigen nur Ströme von bis zu 5mA.
Ich errechnete mir eine mögliche Speicherbelegung von ca. 3120 Datensätzen, bei einem Speicher von 512kb. Warum diese Berechnung praktisch nicht umzusetzen war, wird im Abschnitt Speicher des Kapitels 4.3 Softwareentwicklung (µC), genauer erläutert.

Versuchsaufbauten:

Während der Entwicklungsphase hatte ich mir die Schaltung auf einem Steckbrett aufgebaut, welches mit dem Mikrocontroller-Experimentierboard verbunden war. Der große Nachteil dabei war, dass ich ortsgebunden arbeiten musste, was bei Streckenberechnungen nicht gerade optimal ist.
Doch für die anfänglichen Versuche war das vollkommen ausreichend und bei der Streckenberechnung musste ich mich zumindest vorerst auf die Theorie verlassen. Die grobe Funktionalität konnte ich auch so austesten.

Um dann später doch einen kleinen Testlauf machen zu können, entschied ich mich dazu, die Schaltung auf einer Lochrasterplatine aufzubauen. Mit diesem Aufbau wollte ich hauptsächlich die Geschwindigkeitsmessung des GPS Moduls und das Aufzeichnen einer Strecke testen. Die Anbindung an den PC, die Batterieüberwachung und die „Boost up“ Spannungsversorgung lies ich erst einmal außen vor.
Da die Spannungsversorgung ein wesentlicher Teil einer jeden Schaltung ist, improvisierte ich diese und realisierte sie mit zwei Festspannungsreglern und einem 9V Block. Das sollte zumindest für ein paar Minuten ausreichen. Für die ersten Testläufe war das genug.
Da die Platine des fertigen Projektes jedoch wesentlich kleiner werden sollte musste auch die EMV Beständigkeit, wenn alle Bauteile eng aneinander liegen, getestet werden.
Erst baute ich die Schaltung auf mehreren Platinen auf. Ich entschloss mich dazu, die einzelnen Funktionsgruppen auf getrennten Platinen herzustellen. Somit konnte ich im Falle eines Fehlers diesen besser identifizieren.
Ich trennte also meine Mikrocontrollerschaltung von der USB Anbindung und der

Spannungsversorgung. In Bezug auf die Spannungsversorgung war das Ideal, wie sich zwei Absätze weiter unten zeigen wird, da diese des Öfteren ausgewechselt werden musste.
Als alles fehlerfrei lief, baute ich die komplette Schaltung auf einer Platine auf.

PC-Anbindung:

Nach dem ersten großen Erfolgserlebnis, ging es bald wieder wie gewohnt weiter. Da ich mich gegen die SD Karte entschieden hatte, musste ich mir Gedanken über die Datenübertragung zum PC machen.
Erst wollte ich den Mikrocontroller über die RS232 Schnittstelle und einen USB Konverter an den PC anbinden. Doch das kam mir dann doch ein bisschen umständlich vor, worauf ich im Internet nach anderen Lösungen recherchierte. So bin ich auf den FT232 Controller gestoßen, der direkt UART auf USB konvertiert.

Spannungsversorgung (Teil 2):

Jetzt war es auch langsam an der Zeit, mir ernsthaft Gedanken über die Spannungsversorgung zu machen. Von einem Kollegen erhielt ich ein altes Handy zum ausschlachten. So bekam ich schon mal die Abgriffe für die Kontakte des Akkus. Diese waren glücklicherweise auch mit einem alten Akku der Kamera meiner Freundin kompatibel. Somit war schon einmal gesichert, dass ich den Akku irgendwie einbauen kann, ohne ihn festlöten zu müssen.
Der Akku hatte jetzt aber nur 3,6V. Ich brauchte also einen geeigneten „Boost Konverter“, der mir die 5V erzeugen konnte. Ich bestellte mir erst einen, der laut Datenblatt 130mA bei einer Ausgangsspannung von 5V und einer Eingangs-spannung zwischen 2,2V und 4,2V liefern kann. Meine Schaltung benötigt mit Displaybeleuchtung ca. 110mA. Als ich den „Boost Konverter“ einbaute, lief das Gerät auch vorerst, doch als sich die Displaybeleuchtung anschaltete, brach die Spannung zusammen.
Daraufhin bestellte ich mir einen anderen „Boost Konverter“, der laut Datenblatt 500mA treiben konnte. Dieser fing an unangenehm zu pfeifen und kam bei Belastung ins Schwingen. Das konnte nicht die Lösung sein.
Nach intensiver Suche fand ich einen Baustein, der etwas größer war und auch größere Kondensatoren und Spulen benötigte. Dieser hatte jedoch einen Wirkungsgrad von 96% und konnte bis zu 1A treiben. Ich bestellte ihn, baute ihn ein – er war perfekt!

Batterieüberwachung:

Da bei einem batteriebetriebenen Gerät die Batterieüberwachung wichtig ist, überarbeitete ich nochmals meine bisherige Batterieüberwachung. Diese bestand aus einem Spannungsteiler und dem Analog Digitalwandler des Mikrocontrollers.
Aufgrund der niedrigen Differenz von ca. 0,6V zwischen der Akkuspannung bei geladenem Zustand und bei leerem Zustand, war der Spannungsteiler nicht optimal. Ich wollte zumindest einen Annäherungswert des Akkustandes anzeigen lassen können. Es musste also eine Anpassschaltung her - ein Differenzverstärker.
Bei 3,6V Akkuspannung benötigte ich ca. 0V am AD Eingang und bei 4,2V wollte ich ca. 3,3V. An sich kein Problem, doch verursacht von der PWM des „Boost Konverters“, sind Schwingungen auf der 5V Spannung und der Akkuspannung.
Bei einer großen Differenzspannung sind diese vernachlässigbar, doch bei sinkender Differenzspannung, werden diese zunehmend am Ausgang des Operations-verstärkers bemerkbar. Das hat eine schwankende Akkustatusanzeige zur Folge. Durch eine Glättung der Spannungen könnte dieses Problem gemindert werden. Doch da mir der Annäherungswert ausreichte, beließ ich es dabei!

4.3 Softwareentwicklung (µC)

Grundlagen:

Als erstes informierte ich mich auf der Seite www.kowoma.de über die grundlegenden Eigenschaften der GPS Positionsbestimmung. Dort ist das NMEA 0183 Format mit den unterschiedlichen Datensätzen sehr gut beschrieben. Hier hatte ich auch die Datensätze herausgesucht, welche ich für mein Gerät verwenden wollte.

Filterung der Datensätze:

Als ersten Schritt der Programmierung versuchte ich aus einem seriellen Datenstrom, einzelne Fragmente heraus zu filtern. Dazu schloss ich das Experimentierboard über die RS232 Schnittstelle am PC an und sendete über das Programm „Hyper Terminal“ ASCII-Zeichen.
Als das funktionierte, übertrug ich mit Hilfe eines Simulationsprogramms Datensätze in dem NMEA 0183 Format vom PC zum Experimentierboard und filterte einzelne Informationen heraus.
Jetzt benötigte ich ein echtes GPS-Modul. Nach längerer Internetrecherche bestellte ich das GPS Modul EM-411 von der Firma Navilock.
Um zu sehen wie denn so ein Datenstrom wirklich aussieht, schloss ich das Modul zunächst über den MAX RS232 des Experimentierboards direkt an den PC an und

wertete dort die Daten mit dem „Hyper Terminal“ aus.
Der Weg war jetzt frei das Modul auch an den Mikrocontroller anzuschließen und auszuwerten. Schnell wurde klar, dass die Übertragung mit dem Modul nicht einmal annähernd so stabil ist, wie mit der Simulation.
Ich benötigte also eine bessere Fehlererkennung. Nach längerem Überlegen und schlaflosen Nächten wurde mir erst bewusst, dass ja eine Prüfsumme mit übertragen wird.
Mit dieser Erkenntnis strukturierte ich meinen gesamten Einlesevorgang um, berechnete zu jedem Datensatz die Prüfsumme und verglich sie mit der übertragenen Prüfsumme. So konnte eine ausreichende Qualität der benötigten Daten gewährleistet werden.

Höhe und Geschwindigkeit:

Als ich mir schon fast sicher war das mein Gerät läuft und ich alle Fehler beseitigt hatte, sah ich, mehr oder weniger durch Zufall, dass bei schlechtem Empfang ab und zu in der Anzeige der Höhe und der Geschwindigkeit Hieroglyphen dargestellt werden. Erst konnte ich es nicht glauben, da ich ja die Prüfsumme auswerte und somit wenig Restfehler Möglichkeiten bestehen. Doch es wiederholte sich.
Da es nur bei diesen Werten vorkam, schloss ich daraus, dass es an der 2DFix Positionsbestimmung lag. Ich überarbeitete den Einlesevorgang so, das die Höhe, die Geschwindigkeit und die Anzahl der Satelliten nur bei einer 3DFix Positionsbestimmung ausgewertet und angezeigt werden können. Dadurch wurde dieser Fehler beseitigt.

Abspeicherung von Wegpunkten:

Als die Filterung der Daten aus den Datensätzen ohne Probleme lief, widmete ich mich dem Thema der Abspeicherung der Wegpunkte in einem externen Speicher. Da ich die Speicherung mit einem seriellen I²C EEPROM als Speichermedium realisierten wollte, benötigte ich zunächst einen passenden I²C Treiber.
Einen I²C Treiber für 64kb EEPROMs fand ich recht schnell. Diesen musste ich nun an mein 512kb EEPROM anpassen. Aufgrund des größeren Speichers werden auch mehr Adressen benötigt, wodurch 2Byte für die Adressierung der Speicherzelle übertragen werden müssen und nicht nur eins wie beim 64kb EEPROM.
Ich passte den I²C Treiber an meine Bedürfnisse an und versuchte die Daten abzuspeichern und später über die RS232 Schnittstelle zum PC zu Übertragen. Dort überprüfte ich sie mit dem „Hyper Terminal“. Doch ab einer gewissen Anzahl von Zeichen traten Fehler auf. Es stellte sich heraus, dass ich die physikalischen Speicherseiten des EEPROMs nicht beachtete. Das heißt, wenn ich mehrere Zeichen auf einmal seitenübergreifend abspeicherte, traten Fehler auf. Die physikalischen Speicherseiten sind größer als bei den kleineren EEPROMs. Das musste also zusätzlich im Treiber berücksichtigt werden. Durch die Anpassung der Speicheradressen konnte ich diesen Fehler beheben.
Dadurch ging meine anfangs aufgestellte Rechnung mit den 3120 Datensätzen aber

nicht mehr auf. Da eine physikalische Seite 128Byte groß ist, musste ich darauf achten, dass bei einem Speichervorgang diese Grenze nicht überschritten wird. Anstelle von 21Byte pro Datensatz benötigte ich jetzt 32Byte, wodurch ich nur noch 2048 Datensätze abspeichern konnte.

Streckenberechnung:

Die Standartfunktionen des Gerätes waren schnell erfüllt. Doch von Navigation und Koordinatensystemen hatte ich nicht allzu viel Ahnung. So musste ich mir das Wissen dieser Thematik erst aneignen.
Aufgrund der Tatsache, dass bei einem Track die Koordinaten nur wenige Meter auseinander liegen, musste ich die Entfernung nicht über das sphärische Dreieck berechnen.
Ich rechnete von Hand über den Satz des Pythagoras mit Koordinaten, dessen Entfernung ich kannte. Als das klappte, versuchte ich die Formeln in meinen Quellcode zu übernehmen.
Folgende Formel benutzte ich zur Berechnung der Strecke zwischen den Koordinaten:

$$s = \sqrt{\left(|L1 - L2| \cdot \cos\left(\frac{(B1 + B2) \cdot \pi}{2 \cdot 60 \cdot 180}\right)\right)^2 + |B1 - B2|^2}$$

L1 und L2 = Längenkoordinaten in Minuten.
B1 und B2 = Breitenkoordinaten in Minuten.
s = Strecke

Durch die komplette Umrechnung der Koordinaten in Minuten werden bis zu 4 Stellen vor und nach dem Komma, oder anders gesagt, 8 Nachkommastellen, benötigt. Der Mikrocontroller hat aber nur eine maximale Genauigkeit von 7 Nachkommastellen.
Hier zeigte sich, dass der Mikrocontroller zu ungenau ist und Rundungsfehler entstehen. Dadurch wird die Streckenberechnung verfälscht. Um diesen Fehler zu beheben, hätte ich einen anderen Controller oder eine Menge mehr Aufwand in die Berechnungen stecken müssen.
In Anbetracht mangelnder Zeit und der Tatsache, dass Codespeicher Mangelware ist, belass ich es dabei und widmete mich wieder anderen Aufgaben.

Zeitumstellung:

Da ich über das GPS Modul nur die UTC Weltzeit empfange und ich aber gerne die deutsche Zeit angezeigt haben wollte, überlegte ich, wie das zu realisieren ist. Eine Möglichkeit wäre die Umrechnung in die Winterzeit, plus eine manuelle Umstellung, über einen Menüpunkt, in die Sommerzeit. Doch ich wollte keinen weiteren Menüpunkt in mein schon fertiges Menü zu integrieren und entschied mich, die Umstellung zu automatisieren.

Zu diesem Zeitpunkt wusste ich noch nicht auf was ich mich hier eingelassen hatte. Ich recherchierte im Internet über vielleicht schon vorhandene Lösungen oder Ansätze, doch ich fand nichts Brauchbares. Also informierte ich mich zunächst über die Regeln der Zeitumstellung.
So langsam ahnte ich, dass ich dieses Feature nicht gerade aus dem Ärmel schütteln konnte. Ich wusste erst gar nicht wie ich anfangen sollte, doch jetzt hatte ich mir das schon in den Kopf gesetzt und sah es als Herausforderung an.
Es verging eine ganze Weile bis ich auf die Idee kam, die Tage bis zu einem Referenzdatum zu zählen und so den aktuellen Wochentag herauszubekommen. Dies erwies sich als Schlüssel für die Zeitumstellung.
Mit der Zeitumstellung alleine war es ja dann auch noch nicht getan, Zeiten wie 25 Uhr oder ähnliches mussten ja auch unterbunden werden. Und wenn es laut UTC 23 Uhr ist und es Sommerzeit ist, muss das Datum zusätzlich angepasst werden.
Im Endeffekt bedeutet das alles sehr viel Code für ein mehr oder weniger unauffälliges Feature.

Aufnahmezeit:

Da der Mikrocontroller über keine Taktversorgung verfügt, welche eine präzise Zeitberechnung ermöglicht, habe ich die Berechnung der Aufnahmezeit mit der übertragenen Uhrzeit vom Satelliten realisiert.
Hierbei lagen die zeitintensivsten Punkte bei der Berücksichtigung von Unterbrechungen und der Aufnahme über einen Tag hinaus.

Vertikale Strecke und vertikale Geschwindigkeit:

Die Berechnung der zurückgelegten vertikalen Strecke und der vertikalen Geschwindigkeit sind bei geringen Geschwindigkeiten aufgrund der stark schwankenden Höhenangabe der Satelliten sehr ungenau.
Mit steigender Geschwindigkeit werden diese Werte präziser.

4.4 Softwareentwicklung (PC)

Grundlagen:

Um die Koordinaten später auch in „Google Earth“ darstellen zu können, musste ich mich früher oder später mit der Übertragung der Daten zum PC beschäftigen.
Da wir in der Schule die objektorientierte Programmiersprache „Visual Basic“ angesprochen und mit dieser auch kleine Projekte programmiert haben, entschied ich mich, mein Programm auch in dieser Sprache zu schreiben. Mit dem Visual Studio 2005 entwickelte ich zunächst kleinere Programme.
Zu diesem Zeitpunkt kannte ich den Unterschied zwischen VB6 und VB2005 noch nicht, was sich dann aber schnell änderte. Gerade die serielle Schnittstelle wird bei VB2005 komplett anders bedient als in VB6. Ich brauchte also anderweitig Hilfe und bestellte mir ein Lehrbuch für VB2005. Mit diesem arbeitete ich erst noch einmal die ganzen Grundlagen durch.
Bei der Programmierung der seriellen Schnittstelle konnte mir dieses Buch aber nicht viel weiter helfen, dazu fand ich in der Onlinehilfe von dem Visual Studio gute Anregungen.
Nach mehreren Versuchen hatte ich es geschafft, ein Programm mit VB2005 zu schreiben, welches mir Daten über die serielle Schnittstelle sendet. Das Programm hatte nur einen Haken, ich benötigte jedes Mal den Task Manager um es auch wieder zu beenden.
Auf jeden fall konnte ich so das GPS Modul simulieren und meinem Mikrocontroller einen dauerhaften seriellen Datenstrom liefern.

Zwischen der Hardware und Softwareentwicklung für den µC arbeitete ich immer wieder an meinen Visual Basic Kenntnissen. Als ich mir dann einen gewissen Grundlevel angeeignet hatte, begann ich mit der Programmierung der GPS-Tracker PC-Software.
Die einzige Aufgabe der Software sollte zunächst die Abspeicherung der empfangenen Zeichen in einer Datei sein.

Serielle Schnittstelle:

Die Anbindung der seriellen Schnittstelle an den µC war schnell zu realisieren.
Aber dadurch, dass diese Anwendung anfangs noch auf einem einzigen „Thread“ lief, musste ich im Programm eine Funktion starten, welche die serielle Schnittstelle empfangsbereit machte. Jetzt hatte ich ein, vom „Timeout“ der Schnittstelle eingestelltes Zeitfenster, in dem ich die Daten vom Gerät aus senden musste. Während dieser Zeit gefror zudem noch das Fenster der Anwendung ein, was optisch nicht schön aussieht.
Anfangs war das ausreichend. Ich konnte die empfangenen Koordinaten in einer ASCII-Datei auf meinem Rechner speichern.
Allerdings war auch klar, dass es so nicht bleiben konnte. Also informierte ich mich

über anderweitige Lösungsansätze.
So kam ich auf das Thema Multithreading. Es war ziemlich schwierig geeignete Informationen über die ereignisorientierte Programmierung der seriellen Schnittstelle zu bekommen. Besonders da ich einen asynchronen Zugriff auf die Serielle Schnittstelle realisieren wollte. Der Vorteil hierbei ist, dass sofort ein Hintergrundthread erzeugt wird.
Nach sehr intensiver und langer Internetrecherche kam ich auf die Seite http://www.innovatic.dk. Diese ist zwar auf Englisch, doch genau das was ich brauchte. Ich übertrug das neu erlangte Wissen auf mein Projekt.
Infolgedessen musste ich nicht erst in der Software ein Zeitfenster erzeugen, in dem die Übertragung stattfinden kann. Jetzt war es ausreichend die Daten vom Handgerät aus zu senden, um sie auf dem PC entgegen zu nehmen.
Sporadisch entstanden bei der Übertragung, insbesondere bei größeren Datenmengen, aber auch Fehler. D.h. es gingen teilweise Zeichen verloren.
Erst machte ich das Multithreading dafür verantwortlich. Ich versuchte die Schnittstelle immer weiter zu optimieren und abzusichern, doch alles ohne Erfolg.
Mittlerweile bin ich davon überzeugt, dass es an der Hardware liegt. Wo genau, weiß ich jedoch noch nicht. Es könnte an dem I²C Bus oder auch an der Datenübertragung von oder zu dem USB Controller liegen.
Dadurch, das ich die serielle Schnittstelle des Mikrocontrollers aufgeteilt hatte - die Empfangsleitung ist an das GPS Modul angeschlossen und die Sendeleitung an die USB Schnittstelle - konnte ich keine richtige Kommunikation zwischen dem Gerät und dem PC realisieren. Das heißt, wenn das Gerät die Daten gesendet hat, kann ich diesem nicht mitteilen, ob bei der Übertragung Fehler entstanden sind.
Im Nachhinein würde ich die seriellen Schnittstelle nicht mehr aufteilen. Statt dessen könnte die Emfpangsleitung über einen Schalter oder per Software zwischen dem GPS-Modul und dem USB-Controller geschaltet werden.
Aber als ich das geplant hatte, wollte ich Platz sparen und wusste noch nicht, was alles für Fehler auftreten können.

Konvertierung in das KML-Format:

Jetzt wurde der Aspekt der Konvertierung interessant. Im Internet hatte ich die Programme „GPS-Track-Analyse“ und „GPS Babel“ gefunden. Beide sind in der Lage unterschiedliche Dateiformate von GPS Koordinaten zu konvertieren.
Das Programm „GPS-Track-Analyse“ kann auch ASCII-Dateien in KML-Dateien konvertieren. Mit diesem Programm konvertierte ich meine Koordinaten und ließ sie in „Google Earth“ anzeigen.
Anfangs war das so noch ganz in Ordnung. Doch dann fand ich es zu umständlich, mit meinem Programm die Daten zu Importieren, mit einem fremden Programm zu konvertieren und dann erst in „Google Earth“ anzeigen lassen. Ich wollte versuchen, die Daten selbst zu konvertieren.
Meine Internetrecherche brachte keine zufrieden stellenden Ergebnisse, doch dann kam mir die Idee eine KML-Datei mit einem Texteditor zu öffnen. Und siehe da, ich hatte den Quelltext, der meine Koordinaten in „Google Earth“ einbinden kann. Jetzt musste ich diesen Code nur noch interpretieren, und mein Problem war gelöst – ich konnte meine Daten selbst konvertieren.

Optimierung:

Als die Grundsoftware stand, fielen mir immer mehr Features ein die ich integrieren wollte.
Erst ließ ich, beim markieren einer Datei aus der Liste, das Datum der Erzeugung anzeigen. Das war noch recht einfach, dazu gibt es schon eine fertige Methode.
Später wollte ich dann noch die Anzahl der Wegpunkte und die Länge der Strecke anzeigen lassen. Aufgrund des Speichermangels wollte ich die Werte jnicht über die Schnittstelle übertragen.
Das Problem war die Speicherung. Eine Möglichkeit wäre die Speicherung der Daten in einer speziell angelegten Liste, welche in einer extra Datei gespeichert wird.
Aufgrund der Übersichtlichkeit und dem Handling entschied ich mich aber dagegen. Da ich keine neuen Dateien anlegen wollte, blieb nicht mehr all zu viel. Als ich mit dem KML Format experimentierte und damit auch immer besser klar kam, entschied ich mich dazu, die Werte in den KML-Dateien zu integrieren.
Sobald eine Datei markiert wird, wird diese auf Schlüsselwörter durchsucht. Um die Anzahl der Wegpunkte anzeigen zu lassen, wird das Wort „Latitude“ gezählt. Dieses Wort kommt pro Datensatz einmal in der Datei vor.
Um die Länge der zurückgelegten Strecke anzeigen zu lassen, musste bei der Erstellung der KML-Datei die neu berechnete Strecke zusätzlich abgespeichert werden. Wird jetzt eine Datei markiert, wird diese nach dem Schlüsselwort „Strecke“ durchsucht. Die ersten 5 Zeichen nach dem Wort „Strecke “ inklusive Leerzeichen danach, werden angezeigt.
Somit ergibt sich eine Genauigkeit, welche von der Länge abhängig ist. Wenn über 100 km aufgezeichnet wurden wird der Wert mit einer Nachkommastelle angezeigt. Bei einer Strecke von 1 km, werden 3 Nachkommastellen angezeigt.
Da der Benutzer den Port für die virtuelle Schnittstelle frei wählen können soll, wollte ich die Schnittstelle nicht schon im Quelltext festlegen. Ich wollte aber auch nicht, dass diese bei jedem Start erneut eingestellt werden muss.
Im Visual Studio stieß ich auf die Klasse „Settings“. Das ist eine speziell zur Speicherung von Werten existierende Klasse. Nach kurzer Einarbeitung konnte ich dann den eingestellten Port speichern.
Da mir diese Klasse neue Möglichkeiten gab, entschied ich mich dazu, beim ersten Start der Anwendung den Speicherort für die KML-Dateien abzufragen und über die Klasse „Settings“ zu speichern.
In Hinsicht auf die Fehler bei der Übertragung hatte ich noch keine Lösung gefunden. Aber das die Übertragung bei einem Fehler abgebrochen wird und neu gestartet werden muss, gefiel mir gar nicht.
Diesbezüglich kam ich mit Herr Groß ins Gespräch, worauf ich mich entschied, die Koordinaten zu verwerfen, die einen Fehler enthalten. Somit war gesichert, dass die Übertragung stattfindet. Wegpunkte mit einem Übertragungsfehler werden dadurch nicht übernommen.
Die Anzahl der verworfenen Wegpunkte ist in der KML-Datei gespeichert. Der Benutzer kann sie durch einen Doppelklick auf den Listeneintrag in der GPS-Tracker Software abrufen.

Kapitel 5
Technische Beschreibung

- **Grundlagen**
- **Hardware**
- **Software (µC)**
- **Software (PC)**

=> Wie funktioniert der GPS-Tracker?

5.1 Grundlagen

5.1.1 NMEA 0183 Format

NMEA 0183 ist das Format, in dem die GPS-Daten der Satelliten von dem GPS-Modul ausgegeben werden. Es besteht aus mehreren Datensätzen, welche unterschiedliche Informationen enthalten. Die Informationen werden als ASCII Zeichen übertragen und können somit einfach ausgewertet werden.
Jeder Beginn eines Datensatzes wird mit dem '$' Zeichen gekennzeichnet. Nachfolgend kommt die Senderkennung, z.B. mit den Zeichen „GP" und die Datensatzbezeichnung, bestehend aus drei Zeichen, z.B. „RMC".
Dem entsprechend sieht eine vollständige Kennzeichnung eines Datensatzes z.B. so aus: „$GPRMC".
Das Ende des jeweiligen Datensatzes wird mit dem '*' Zeichen gekennzeichnet. Nach dem '*' wird zusätzlich eine Prüfsumme übertragen. Diese besteht aus zwei Ziffern und ist hexadezimal codiert.
Die einzelnen Informationen, welche ein Datensatz enthält, werden durch Kommas getrennt. Wird eine Information aus irgendwelchen Gründen nicht übertragen, wird sie einfach weggelassen. Doch die dazugehörigen Kommas werden ohne Leerzeichen beibehalten. Durch das Zählen der Kommas, kann ein Empfänger aus jedem Satz die entsprechenden Informationen richtig zuordnen. Jeder Datensatz hat eine definierte Anzahl von Kommas.
Bei Empfang werden ständig abwechselnd Datensätze empfangen.

Die wichtigsten Datensätze werden nachfolgend kurz erläutert.

$GPRMC Datensatz:

$GPRMC,191410,A,4735.5634,N,00739.3538,E,0.0,0.0,181102,0.4,E,A*19

$	Satzanfang
GP	Senderkennung
RMC	Datensatzkennung
191410	Uhrzeit der Positionsbestimmung: 19:14:10 UTC-Zeit
A	Empfängerwarnung, A = Daten OK, V = Warnung
47	Breitengrad
35.5634	Breitenminute
N	N = nördliche Breite, S = südliche Breite
007	Längengrad
39.3538	Längenminute
E	E = östliche Länge, W = westliche Länge
181102	Datum der Positionsbestimmung: 18.11.02
*	Satzende
19	Prüfsumme: 19hex

$GPGGA Datensatz:

$GPGGA,191410,4735.5634,N,00739.3538,E,1,04,4.4,351.5,M,48.0,M,,*45

$	Satzanfang
GP	Senderkennung
GGA	Datensatzkennung
04	Anzahl der erfassten Satelliten = 4
351.5	Höhe über Meer (über Geoid) in Metern = 351.5m
*	Satzende
45	Prüfsumme: 45hex

$GPGSA Datensatz:

$GPGSA,A,3,,,,15,17,18,23,,,,,,4.7,4.4,1.5*3F

$	Satzanfang
GP	Senderkennung
GSA	Datensatzkennung
A	Art der Positionsbestimmung: 3 = 3D-fix, 2 = 2D-fix, 1 = kein Fix
,,,,,,	An dieser Stelle ist gut zu erkennen, wie reservierte Datenfelder nicht übertragen wurden, und nur die Kommas beibehalten werden
4.7	PDOP (allgemeine Genauigkeit)
4.4	HDOP (horizontale Genauigkeit)
1.5	VDOP (vertikale Genauigkeit)
*	Satzende
3F	Prüfsumme: 3Fhex

Die „DOP“ Genauigkeitswerte geben an, wie gut die geometrische Ausrichtung der Satelliten für die Positionsbestimmung ist. Desto kleiner der Wert, desto genauer kann die Position bestimmt werden. Ein ganz guter Wert ist z.B. 3.

$GPVTG Datensatz:

$GPVTG,0.0,T,359.6,M,0.0,N,0.0,K*47

$	Satzanfang
GP	Senderkennung
VTG	Datensatzkennung
0.0	Geschwindigkeit über Grund in km/h
*	Satzende
47	Prüfsumme: 47hex

5.1.2 Keyhole Markup Language

Bei KML handelt es sich um ein Dateiformat zur Darstellung geografischer Daten in einem Earth-Browser wie beispielsweise Google Earth, Google Maps und Google Maps für Handys.
Eine KML-Datei wird weitgehend wie eine HTML- (oder XML-)Datei in einem Webbrowser ausgewertet. Wie HTML besitzt auch KML eine tag-basierte Struktur mit Namen und Attributen für spezielle Darstellungen. Google Earth und Google Maps fungieren sozusagen als Browser für KML-Dateien.

Zur Markierung einer Koordinate in Google Earth reicht es aus die folgende Textpassage in einem Texteditor als *.kml Datei abzuspeichern und zu öffnen:

```
<Folder>
<name>Trackpoints</name>
<Placemark>
<name> TP0</name>
<Snippet maxLines="2"></Snippet>
<description><![CDATA[<table>
<tr><td><b>Zeitpunkt:</b> 05.02.2008 21:35:17</td></tr>
<tr><td><b>Geschwindigkeit:</b> 0.6 km/h </td></tr>
<tr><td><b>Longitude:</b> 9.17194833333333</td></tr>
<tr><td><b>Latitude:</b> 47.6735766666667</td></tr>
<tr><td><b>Höhe:</b> 418.5 m </td></tr>
</table>]]></description>
<styleUrl>#Trackpoint</styleUrl>
<Point>
<altitudeMode>clampToGround</altitudeMode>
<coordinates>9.17194833333333,47.6735766666667</coordinates>
</Point>
</Placemark>
</Folder>
```

Um einen ganzen Track einzubinden bedarf es etwas mehr Quellcode. Falls dazu Interesse bestehen sollte, kann eine von der GPS-Tracker Software erzeugte *kml Datei, mit einem beliebigen Texteditor, geöffnet werden. In diesem wird dann der Quellcode dargestellt.
Eine ausfühliche Beschreibung ist auch unter der Website http://code.google.com/apis/kml/documentation/ erhältlich.

5.2 Hardware

5.2.1 Blockschaltbild

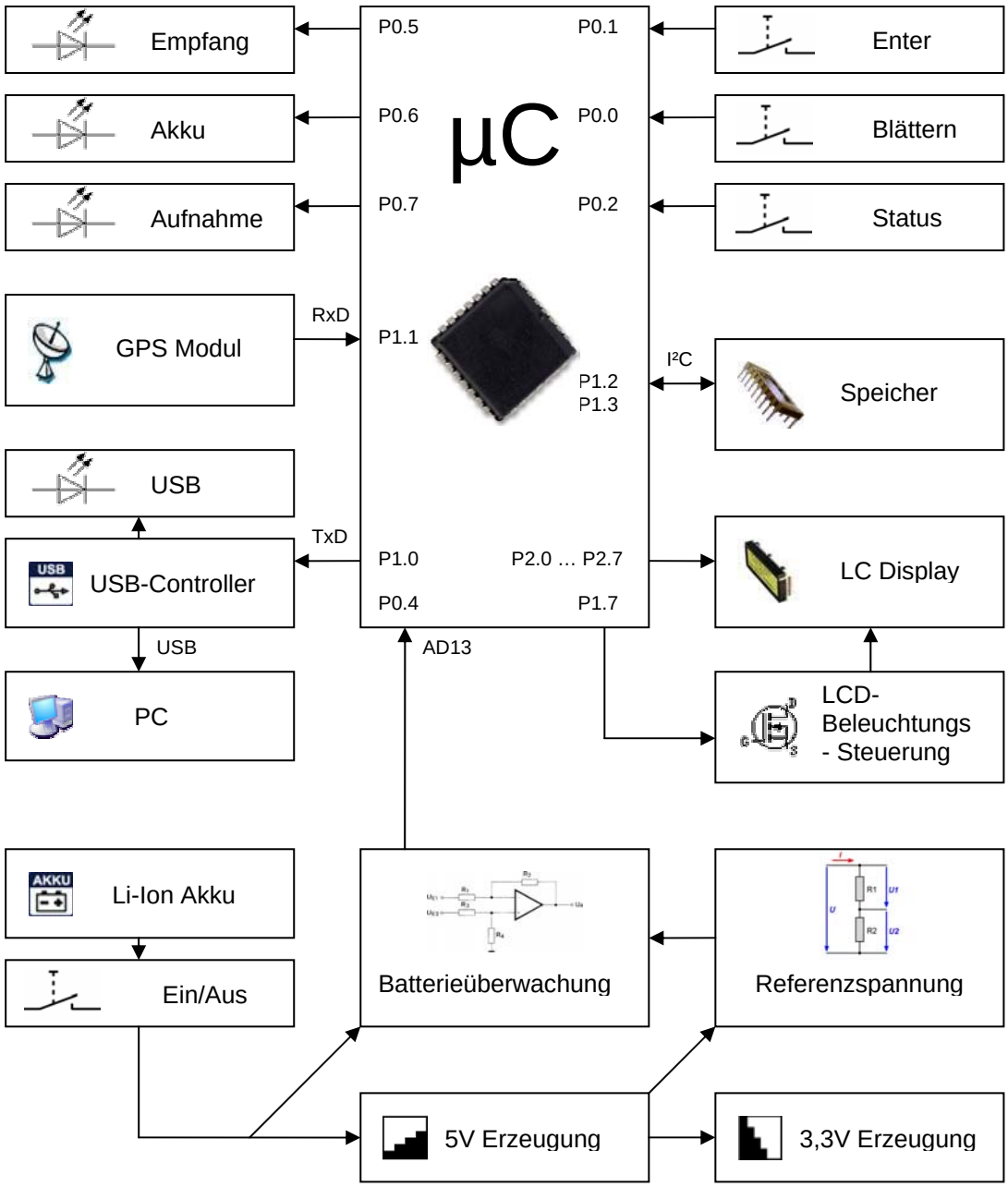
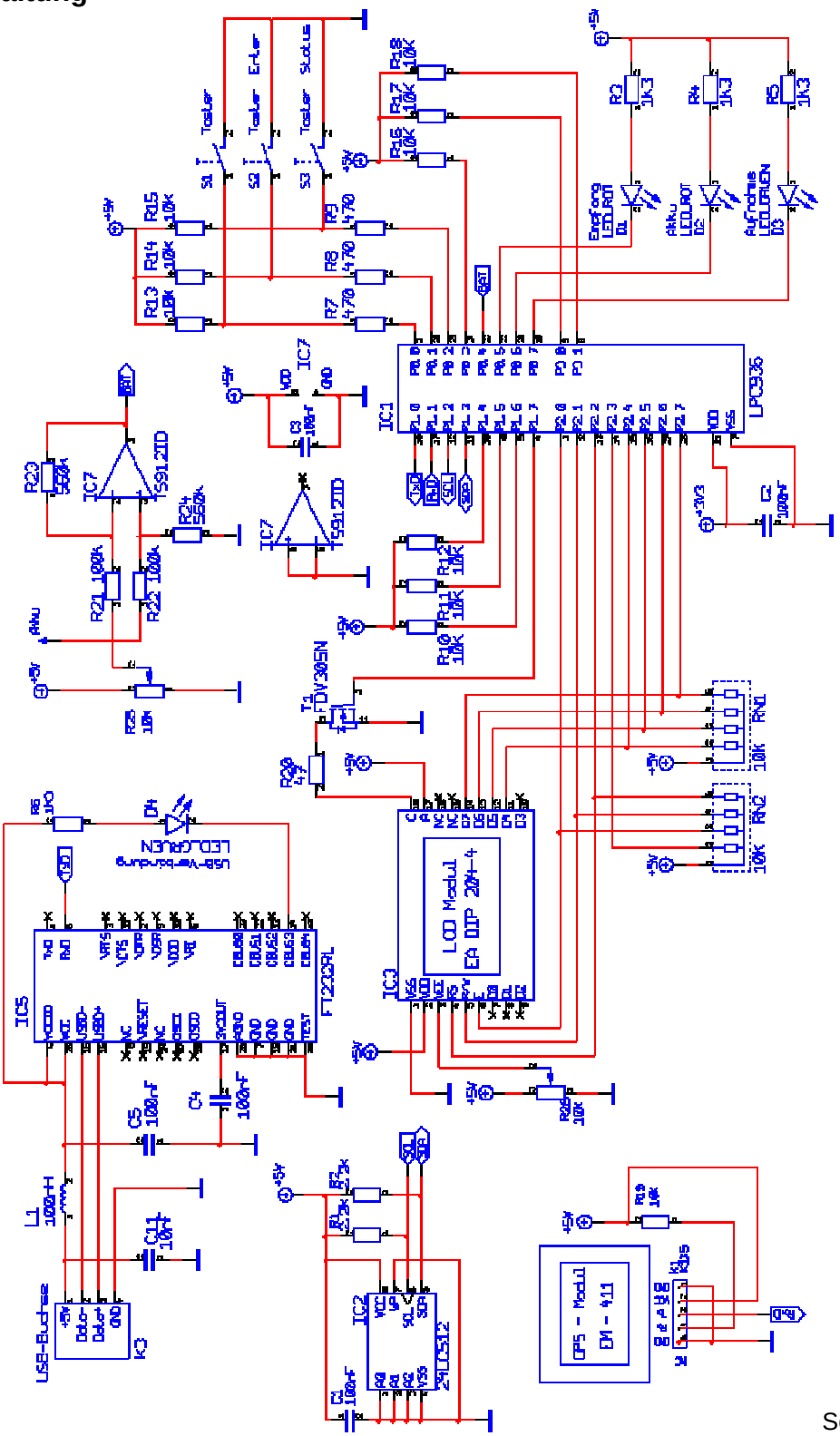


Bild 5

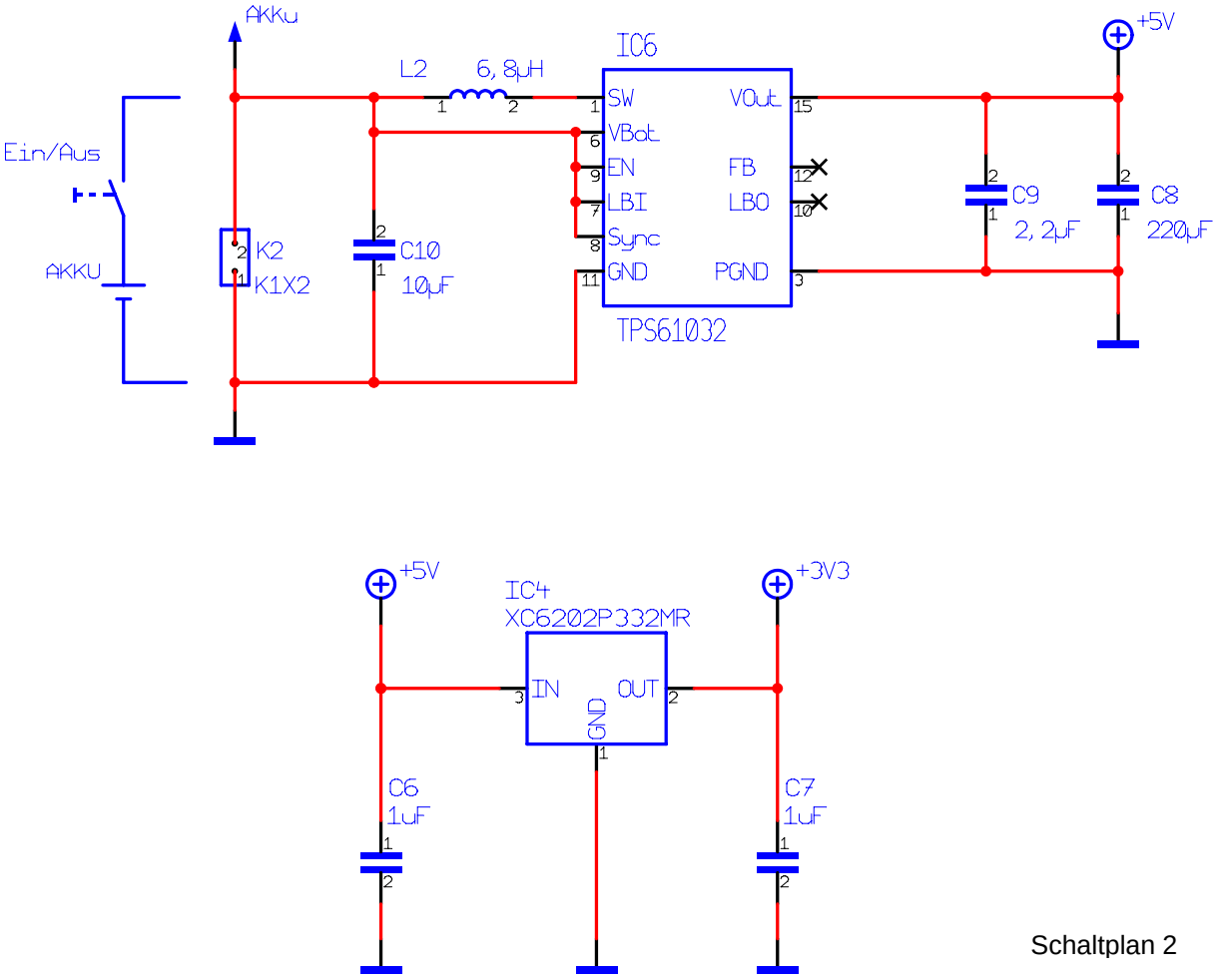
5.2.2 Schaltplan

Hauptschaltung



Schaltplan 1

Spannungsversorgung

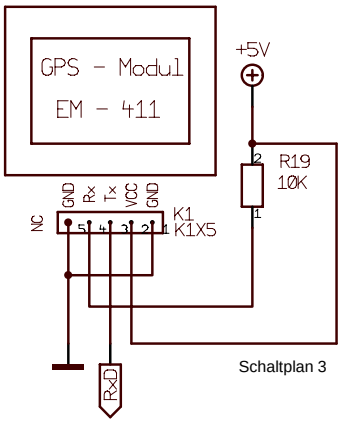


Schaltplan 2

5.2.3 Schaltplan Erläuterungen

GPS-Modul EM-411:

Das GPS-Modul liefert einen TTL kompatiblen Datenstrom mit einer Baudrate von 4800 Baud. Die Daten werden als ASCII Zeichen übertragen und können direkt über die serielle Schnittstelle des Mikrocontrollers (UART) eingelesen werden.

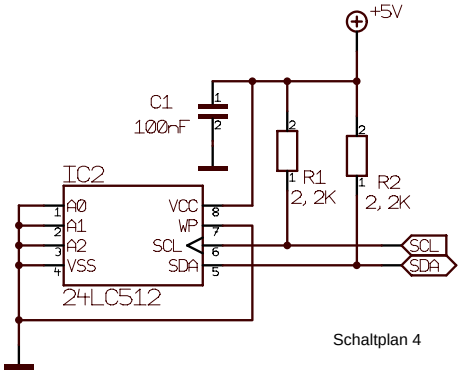


Schaltplan 3

Über den Tx Ausgang des GPS-Moduls werden die Daten der Satelliten über den RxD Eingang des Mikrocontrollers eingelesen.
Um Störungen zu vermeiden wird der Rx Eingang des GPS-Moduls auf High gelegt.
Das Modul benötigt eine Versorgungsspannung von 5V und hat eine Stromaufnahme von ca. 60mA.

EEPROM 24LC512:

Das externe EEPROM hat eine Speichergröße von 512kBit und wird zur Speicherung der Wegpunkte benutzt. Pro Datensatz werden 32Byte benötigt, wodurch sich eine maximale Speicherung von 2048 Datensätzen ergibt. Die Ansteuerung erfolgt über den I²C-Bus.

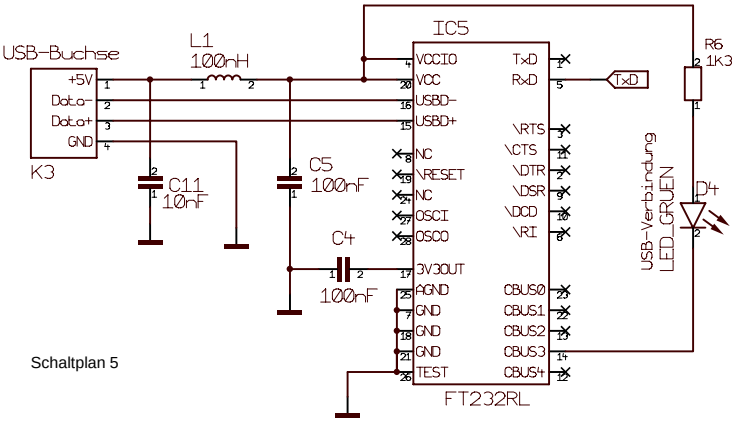


Schaltplan 4

Hardwaremäßig wurde dem Speicher, über die Eingänge A0, A1 und A2, die Adresse 0 zugeordnet.
Über die SCL Leitung wird der Takt für die serielle Datenübertragung geliefert und über die SDA Leitung werden die Daten übertragen. Diese beiden Leitungen sind auch direkt am I²C Interface des µC angeschlossen
R1 und R2 sind „pull up“ Widerstände, die für saubere Pegel an den Eingängen sorgen.
C1 ist ein Abblockkondensator, er glättet die Betriebsspannung bei kleinen Schwankungen.

USB Anbindung über den FT232:

Über den USB Controller FT232 können die Signale der UART des µC direkt in USB Signale umgewandelt und über ein Kabel zum PC übertragen werden.

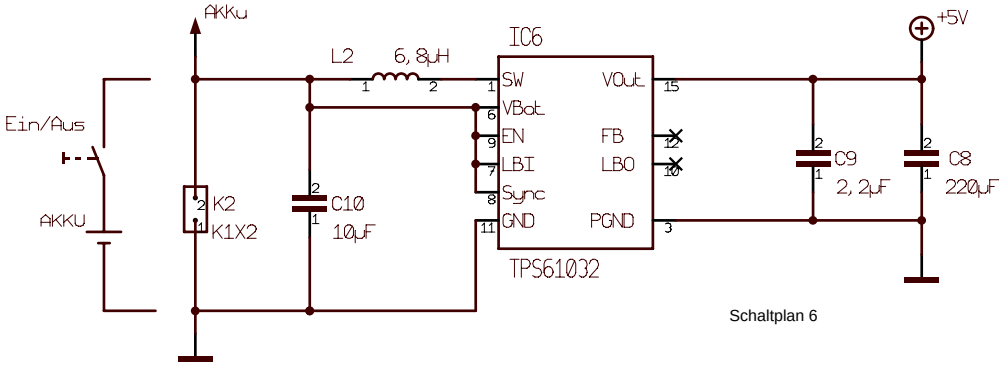


Schaltplan 5

Die Schaltung wird über die USB Schnittstelle des PCs mit Spannung versorgt.
Die LED D4 leuchtet sobald das Gerät mit dem PC verbunden ist.
Der RxD Eingang des USB Controllers ist direkt mit dem TxD Ausgang des µC verbunden.
Die Induktivität L1 glättet Span - nungsspitzen der Versorgungs- spannung und die Kondensatoren C5 und C11 glätten kleinere Spannungseinbrüche.
Der Ausgang 3V3OUT wird nach den Vorgaben des Datenblattes über einen Kondensator mit Ground verbunden.

Spannungsversorgung:

Die Spannungsversorgung besteht aus zwei Teilen, der 5V und der 3,3V Erzeugung.
Aus der Akkuspannung (3,6V – 4,2V) wird über den „Boost Konverter“ TPS61032 die 5V Spannungsversorgung zur Verfügung gestellt. Der Wirkungsgrad bei dieser Schaltung liegt bei etwa 96%.



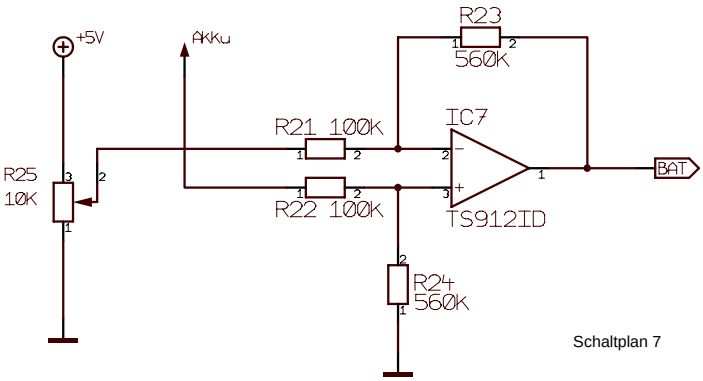
Schaltplan 6

Mit Hilfe der Spule L2 und dem Kondensator C10 wird eine Puls-Weiten-Modulation erzeugt. Über diese Pulse werden die 5V generiert.
Die Kondensatoren C8 und C9 glätten die Ausgangsspannung.

Da der Mikrocontroller eine Betriebsspannung von 3,3V benötigt, werden aus den zuvor erzeugten 5V, über einen Festspannungsregler, 3,3V zur Verfügung gestellt.

Batterieüberwachung:

Die Batterieüberwachung wurde mit einem Differenzverstärker als Anpassschaltung für die Spannungspegel des Akkus an den AD-Eingang des µC realisiert.



Schaltplan 7

Über den Differenzverstärker wird die Akkuspannung mit einer Spannung von 3,6V verglichen. Die 3,6V werden durch den Trimmer R25 auf den Eingang gegeben.
Der Ausgang des Differenzverstärkers wird direkt auf einen AD-Eingang des Mikrocontrollers gelegt.
Durch die Widerstände R21 bis R24 wurde ein Verstärkungsfaktor von 5,6 eingestellt. Somit ergibt sich bei einer Akkuspannung von 3,6V, eine Ausgangsspannung von etwa 0V am Analog Digitaleingang. Bei einer Akkuspannung von 4,2V liegen am AD-Eingang des µC ca. 3,3V an.
Der Akkustand kann so theoretisch mit einer Auflösung von ca. 2,5mV bestimmt werden.

Um die Schaltung abzugleichen, muss über den Trimmer R25 die Spannung am invertierenden Eingang (M1) der Schaltung eingestellt werden. Die Schaltung ist abgeglichen, wenn 3,6V (die Spannung des Akkus bei leerem Zustand) am Eingang anliegt.

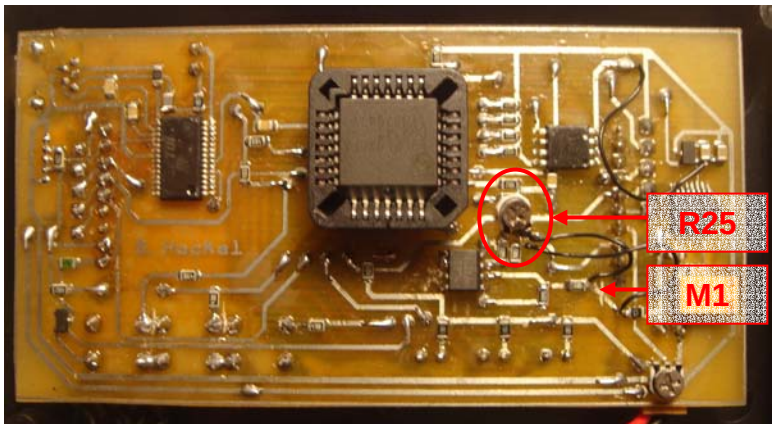


Bild 6

5.2.4 Technische Daten

	min.	**max.**
Spannungsversorgung	3,5V	4,5V
Stromaufnahme	130mA	170mA
Akkulaufzeit		4h
Max. Anzahl der Wegpunkte		2047
Max. Aufzeichnungsdauer		2h 50min

5.2.5 Platinenlayout

Kupfer beidseitig (Bemaßung in mm)

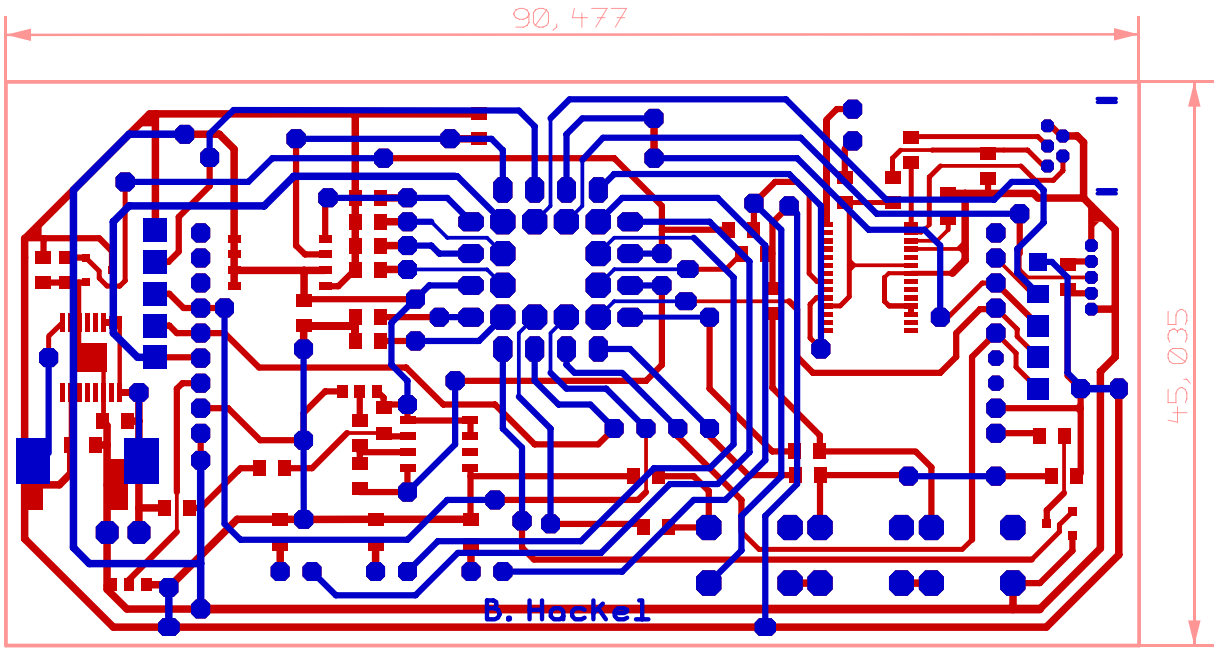


Bild 7

Bestückung oben

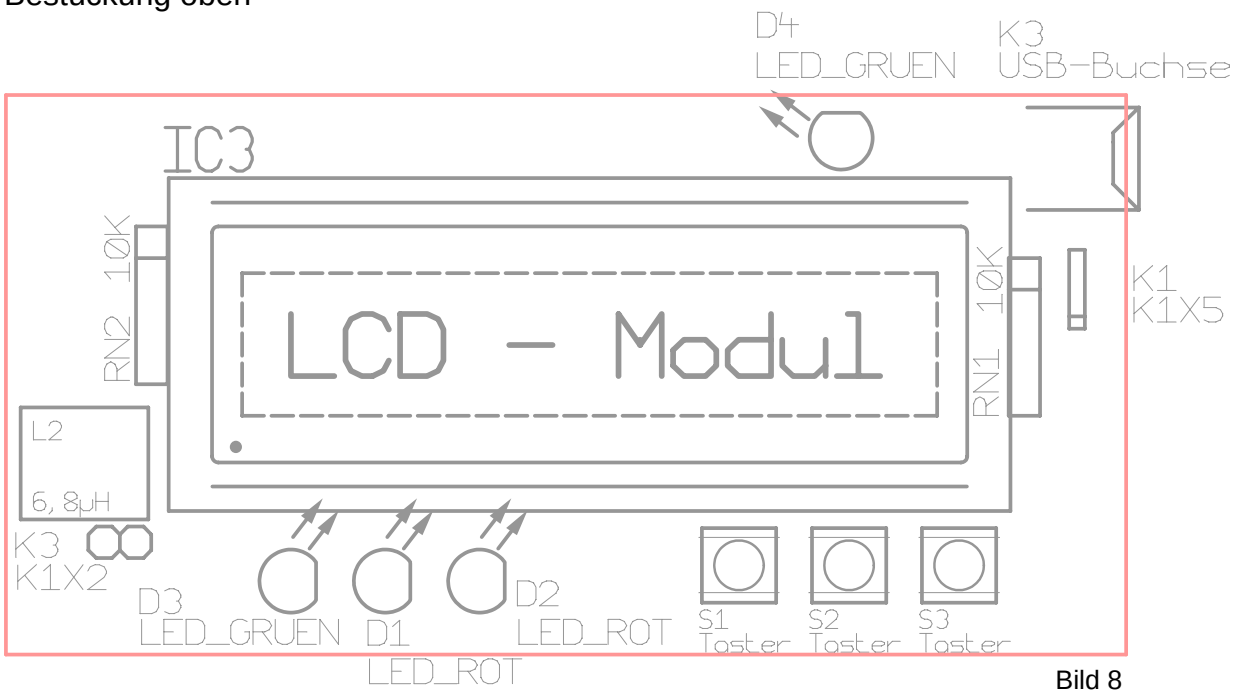


Bild 8

Bestückung unten

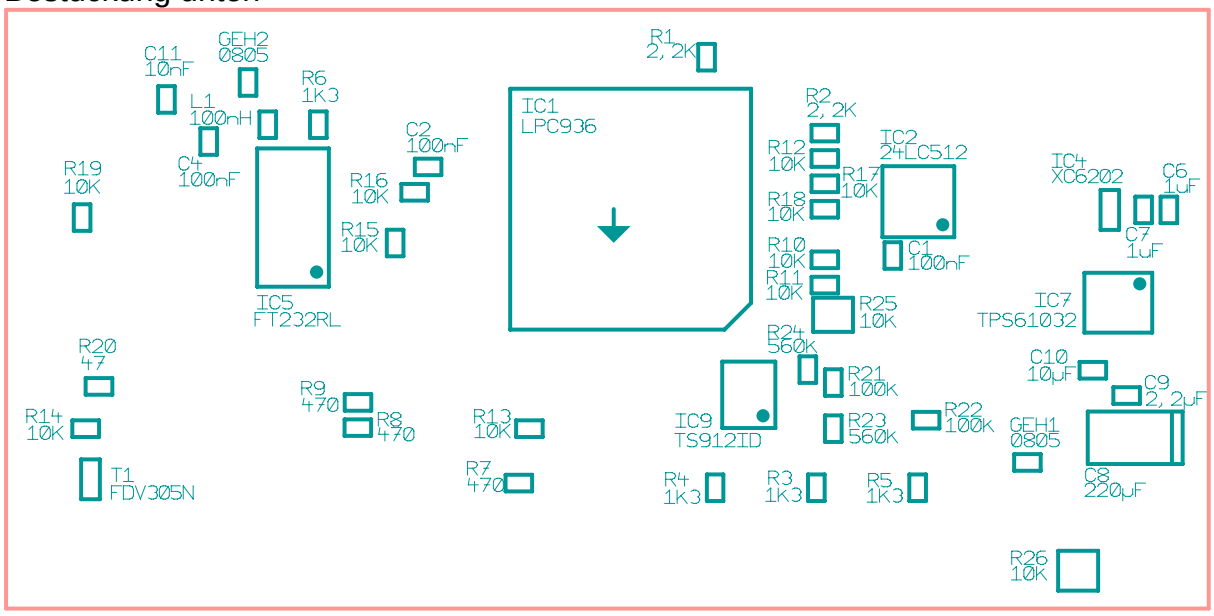


Bild 9

5.2.6 Stückliste

Bauteil	Name	Wert / Bezeichnung	Anz.	Einzelpreis	Preis Netto
GPS - Modul		Navilock EM 411	1	38,43 €	38,43 €
Mikrocontroller	IC1	LPC936	1	4,17 €	4,17 €
EEPROM	IC2	24LC512	1	3,40 €	3,40 €
LCD - Modul	IC3	DIP204-4NLED	1	22,76 €	22,76 €
Spannungsregler	IC4	XC6202P332MR	1	0,88 €	0,88 €
USB Controller	IC5	FT232RL	1	3,15 €	3,15 €
Boost Konverter	IC6	TPS61032	1	3,20 €	3,20 €
Operationsverstärker	IC7	TS912ID	1	0,61 €	0,61 €
Feldeffektransistor	T1	FDV305N	1	0,28 €	0,28 €
Low Current LED	D1, D2	Rot 3mm	2	0,09 €	0,18 €
Low Current LED	D3, D4	Grün 3mm	2	0,09 €	0,18 €
Kondensator	C1 - C5	100nF	5	0,08 €	0,42 €
Kondensator	C6, C7	1µF	2	0,31 €	0,62 €
Kondensator	C8	220µF	1	3,00 €	3,00 €
Kondensator	C9	2,2µF	1	0,14 €	0,14 €
Kondensator	C10	10µF	1	0,32 €	0,32 €
Kondensator	C11	10nF	1	0,04 €	0,04 €
Induktivität	L1	100nH	1	0,22 €	0,22 €
Induktivität	L2	6,8µH	1	1,98 €	1,98 €
Widerstand	R1, R2	2,2kΩ	2	0,09 €	0,17 €
Widerstand	R3 - R6	1,3kΩ	4	0,03 €	0,12 €
Widerstand	R7 - R9	470Ω	3	0,07 €	0,21 €
Widerstand	R10 - R19	10kΩ	10	0,07 €	0,69 €
Widerstand	R20	47Ω	1	0,21 €	0,21 €
Widerstand	R21, R22	100kΩ	2	0,07 €	0,14 €
Widerstand	R23, R24	560kΩ	2	0,07 €	0,14 €
Trimmer	R25, R26	10kΩ	2	0,72 €	1,45 €
Widerstandsnetzwerk	RN1, RN2	10kΩ	2	0,07 €	0,14 €
Kurzhubtaster	S1 - S3	9,5mm	3	0,19 €	0,57 €
Schiebeschalter		Miniatur	1	0,20 €	0,20 €
Sockel		PLCC 28	1	0,28 €	0,28 €
Brücke	GEH1 + 2	0Ω	2	0,03 €	0,06 €
Platinensteckverbinder	K1	5 polig	1	0,31 €	0,31 €
Platinensteckverbinder	K2	2 polig	1	0,25 €	0,25 €
USB-Buchse	K3	USB Mini	1	0,42 €	0,42 €
USB Kabel		USB Mini	1	0,83 €	0,83 €
Akku		NP-FT1	1	8,36 €	8,36 €
Platine		Doppelseitig	1	2,52 €	2,52 €
Gehäuse		102 x 61 x 26mm	1	1,85 €	1,85 €
				Summe Netto	**102,88 €**
				+19% Mwst.	19,55 €
				Summe Brutto	**122,43 €**

Tabelle 2

5.3 Software (µC)

5.3.1 Allgemeines:

Für die Entwicklung der Software wurde die IDE „µVision3“ von der Firma Keil in Kombination mit dem LPC Experimentierboard und dem Emulatorboard EPM900, ebenfalls von der Firma Keil, benutzt.

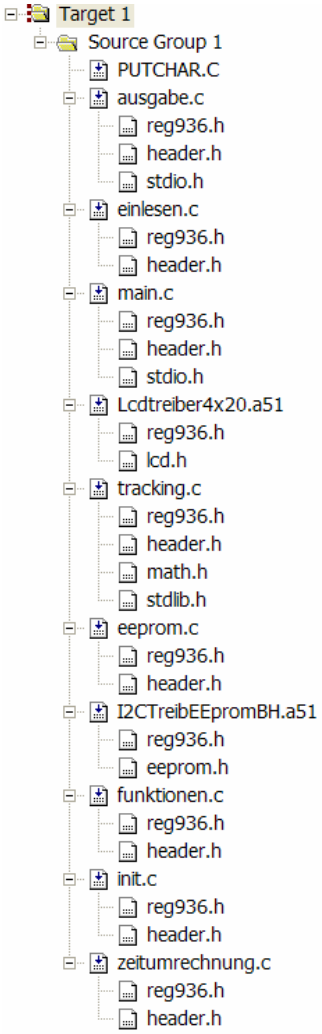


Bild 10

Die Software ist bis auf die Treiber komplett in der Programmiersprache C geschrieben. Die Treiber sind in Assembler geschrieben und an die Hardware angepasst worden.

Für die Software auf dem Mikrocontroller wird ein Code-Speicher von 15255Byte benötigt. Des Weiteren werden im Data-Speicher 105Byte und im Xdata-Speicher 327Byte belegt.

Links ist die Struktur des Programms zu sehen. Die Funktionen und Aufgaben des Programms sind in einzelnen Dateien abgelegt. Dies hat den Vorteil einer besseren Übersicht und somit einer effektiveren Programmierung.

Die Datei „header.h“ ist ein selbst geschriebener Header. In diesem werden die Variablen und Funktionsprototypen für die anderen Dateien zugänglich gemacht, sowie die Taster definiert.
Durch die Einbindung der Header „math.h“, „stdio.h“ und „stdlib.h“ konnten fertige Funktionen, wie die cosinus Funktion oder die ASCII in Float Konvertierung benutzen werden.
Mittels der Header „eprom.h“ und „lcd.h“, wurden die Funktionen der jeweiligen Treiber den restlichen Modulen bekannt gemacht.
Die Datei reg936.h erlaubt die Benutzung der prozessorspezifischen Flag- und Adressbezeichnungen.

Auf den folgenden Seiten wird auf die einzelnen Module genauer eingegangen. Aufgrund der Komplexität des Programms wird jedoch nicht jedes Detail ausführlich beschrieben.
Falls die detaillierte Lösung eines Moduls oder einer Funktion gewünscht ist, kann diese gerne im Quellcode auf der beiliegenden CD angesehen werden.

Code Komprimierung

Um den oben die Software überhaupt auf den Mikrokontroller zu bekommen muss diese komprimiert werden. Ansonsten hat die Software einen Speicherbedarf von 16410Byte und passt somit nicht in den zur Verfügung gestellten Codespeicher des Mikrocontrollers.
Die Software wird komprimiert indem folgende Einstellung in der IDE, unter „Options for Target“ vorgenommen wird:

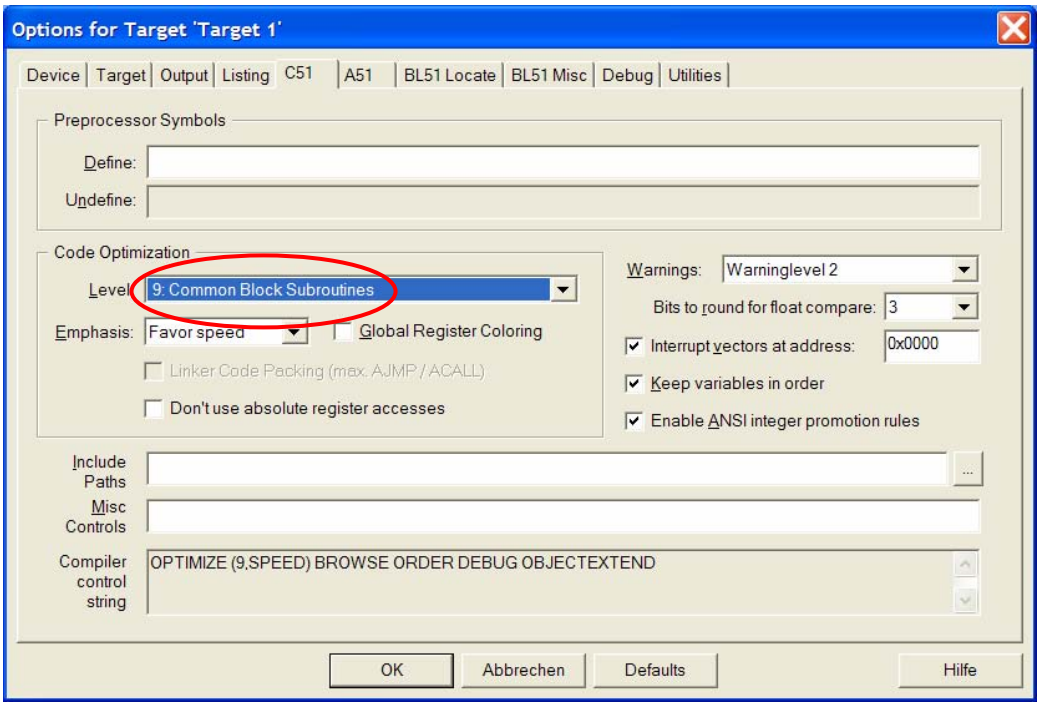


Bild 11

Pinbelegung des Mikrocontrollers

Portnummer	Bezeichnung	Betriebsart	Bezeichnung
P0.0	TASTER1	Input-Only	Blättern
P0.1	TASTER2	Input-Only	Enter
P0.2	TASTER3	Input-Only	Status
P0.3	AD12	Input-Only	A/D-Wandler Eingang
P1.0	TxD	Quasi Bidirektional	Schnittstelle (Senden)
P1.1	RxD	Quasi Bidirektional	Schnittstelle (Empfang)
P1.2	SCL	Open Drain	I²C Taktleitung
P1.3	SDA	Open Drain	I²C Datenleitung
P1.7	LCD_BEL	Quasi Bidirektional	LCD Beleuchtung
P2.0	E	Quasi Bidirektional	Enable (LCD)
P2.1	R/W	Quasi Bidirektional	Read/Write (LCD)
P2.2	RS	Quasi Bidirektional	Umschaltung Befehl/Daten (LCD)
P2.4	D4	Quasi Bidirektional	Daten (LCD)
P2.5	D5	Quasi Bidirektional	Daten (LCD)
P2.6	D6	Quasi Bidirektional	Daten (LCD)
P2.7	D7	Quasi Bidirektional	Daten (LCD)

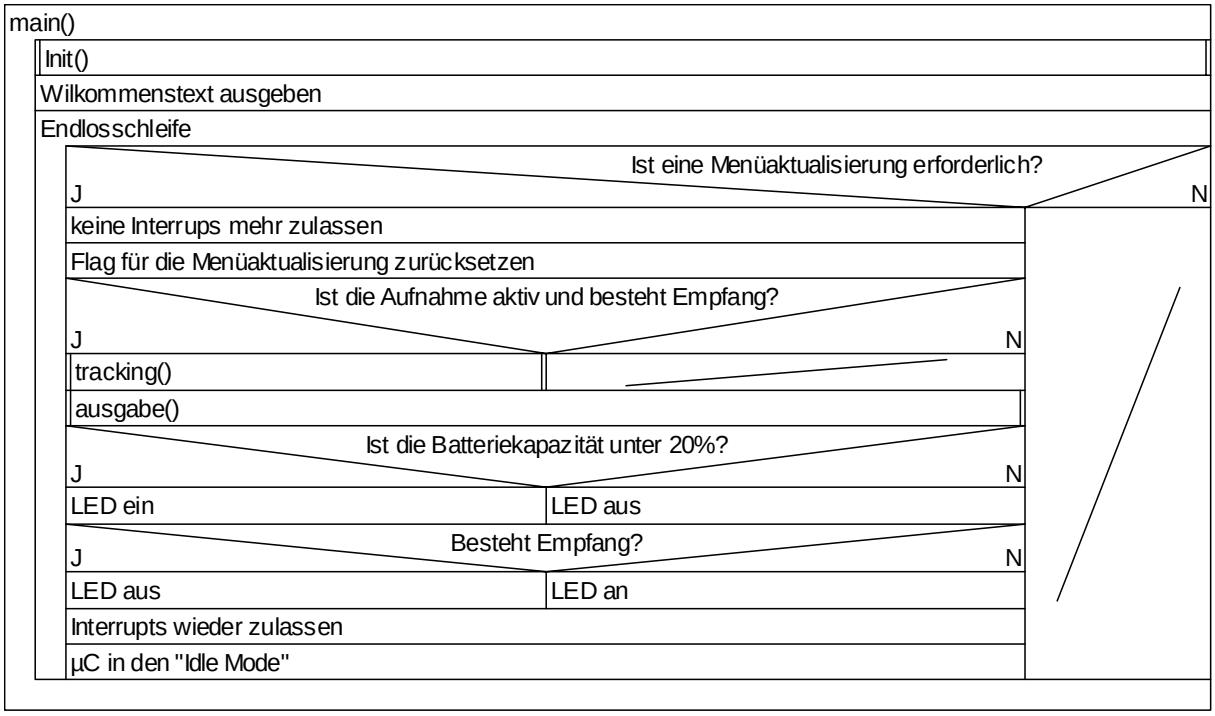
Die übrigen Pins wurden alle mit der Betriebsart „Input only“ konfiguriert.

5.3.2 Der grobe Ablauf

Die komplette Software ist Interrupt gesteuert. Das heißt, der Programmablauf wird hauptsächlich durch Ereignisse gesteuert. Das Hauptprogramm ist eine kleine Endlosschleife. Diese wird nur durchlaufen, wenn ein bestimmtes Flag gesetzt wird. Das Flag wird alle 2 Sekunden durch einen Timer oder durch ein Ereignis, wie z.B. ein Tastendruck gesetzt.
In der Schleife werden zwei Funktionen aufgerufen. Eine Funktion ist für die Displayanzeige zuständig und die andere ist, während einer Aufnahme, für die Berechnungen zuständig. Ansonsten werden hier nur noch die LEDs für die Batterieanzeige und die Empfangsanzeige, gesteuert.
In der übrigen Zeit ist das Programm im „Idle – Mode“. Das ist ein stromsparender Betriebsmodus des Mikrocontrollers. In diesem Modus bleibt der Mikrocontroller bis ein Interrupt ausgelöst wird.
Beim Einschalten des Gerätes werden die gespeicherten Daten aus dem internen EEPROM geladen. Das sind Informationen der letzten aufgezeichneten Strecke. Wenn eine Aufnahme beendet wird, werden alle Informationen die im Trackingmenü abrufbar sind, gespeichert.
Der Einlesevorgang der GPS-Daten findet in der Interrupt Service Routine des seriellen Interrupts statt. Wird ein Zeichen empfangen, wird eine Funktion aufgerufen, welche das empfangene Zeichen der richtigen Variablen oder dem richtigen String zuordnet.
Wird eine Taste gedrückt, wird zuerst analysiert, welche der 3 Tasten gedrückt wurde. Anschließend wird geklärt, an welcher Stelle im Menü die Taste gedrückt wurde. Darauf basierend wird die gewünschte Aktion durchgeführt.

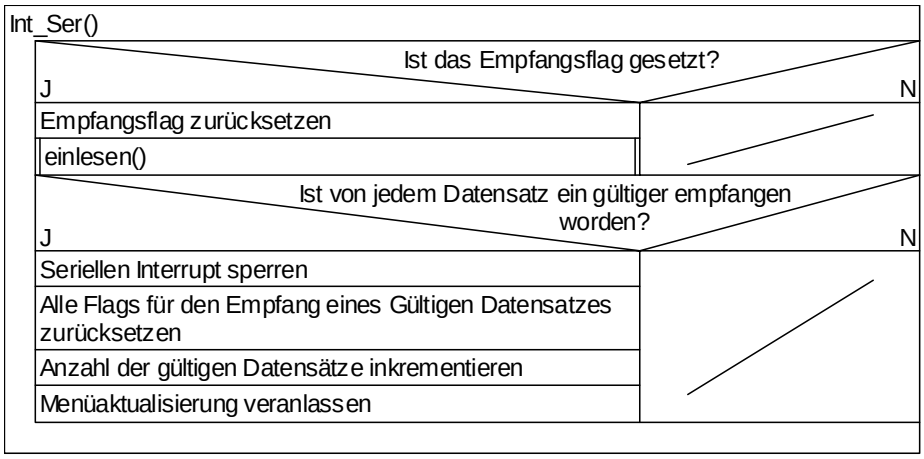
5.3.3 main.c

Funktion: **main**()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Die Funktion main() ist eine Endlosschleife. Hier verweilt der Mikrocontroller während er auf ein Ereignis (Interrupt) wartet. In dieser Wartezeit geht der µC automatisch in den „Idle-Mode“.
In der Endlosschleife finden die Menüaktualisierung und die Berechnungen statt. Diese werden nur alle 2s, oder durch Sonderereignesse veranlasst, abgearbeitet. Sonderereignisse sind z.B. ein Tastendruck oder der Empfang je eines gültigen Datensatzes.
Das vermeidet ein „flackern“ des Displays, verursacht durch zu häufiges aktualisieren und unnötige CPU Belastung, durch zu häufige Berechnung der Trackingdaten.



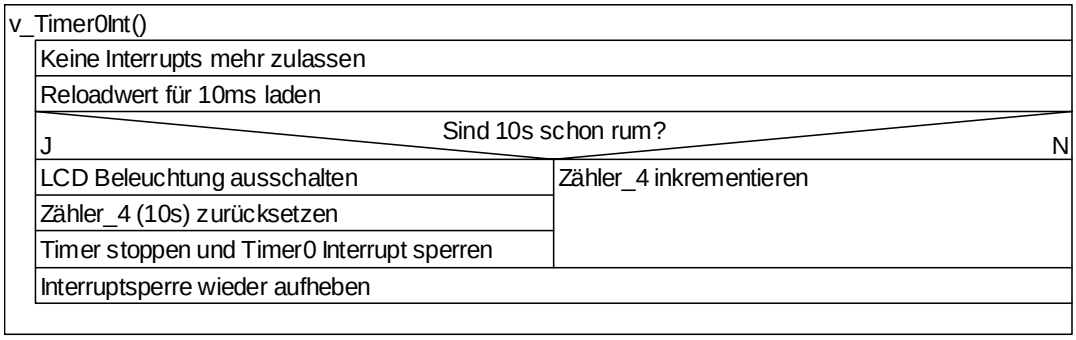
Struktogramm 1

Funktion: **Int_Ser**()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Die Interrupt Service Routine wird bei jedem empfang eines Zeichens abgearbeitet.
Wenn von den insgesamt 4 verschiedenen Datensätzen (RMC, GGA, VTG, GSA) jeweils ein gültiger empfangen wurde, wird der serielle Interrupt gesperrt und erst nach ca. 2s wieder freigegeben.



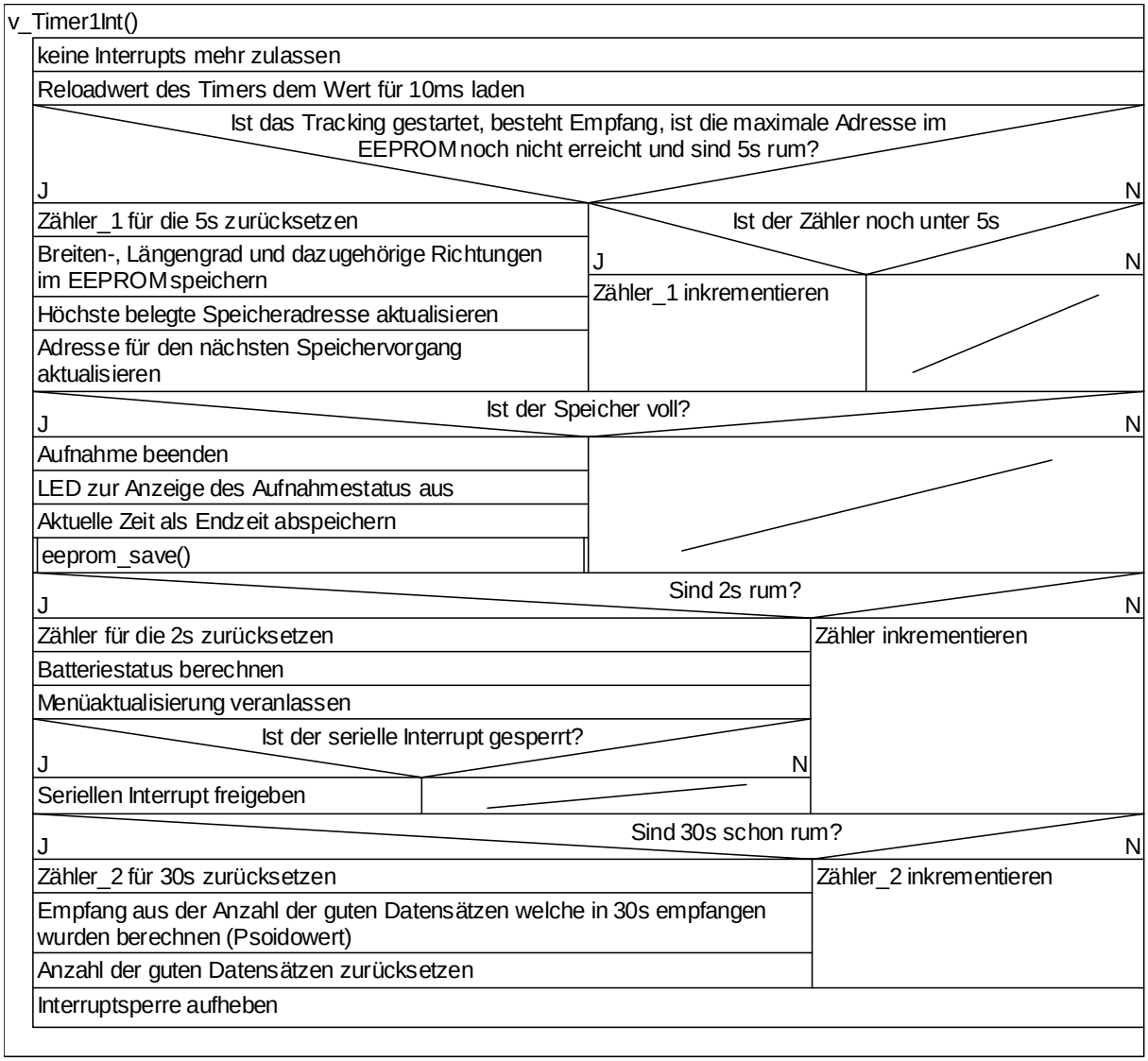
Struktogramm 2

Funktion: **v_Timer0Int**()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Die ISR wird für die Displaybeleuchtung verwendet. Bei jedem Tastendruck wird der Timer0 aktiviert und die Displaybeleuchtung gestartet. Nach Ablauf von 10s wird der Interrupt ausgelöst in dem die Beleuchtung wieder abgeschaltet und der Timer deaktiviert wird.



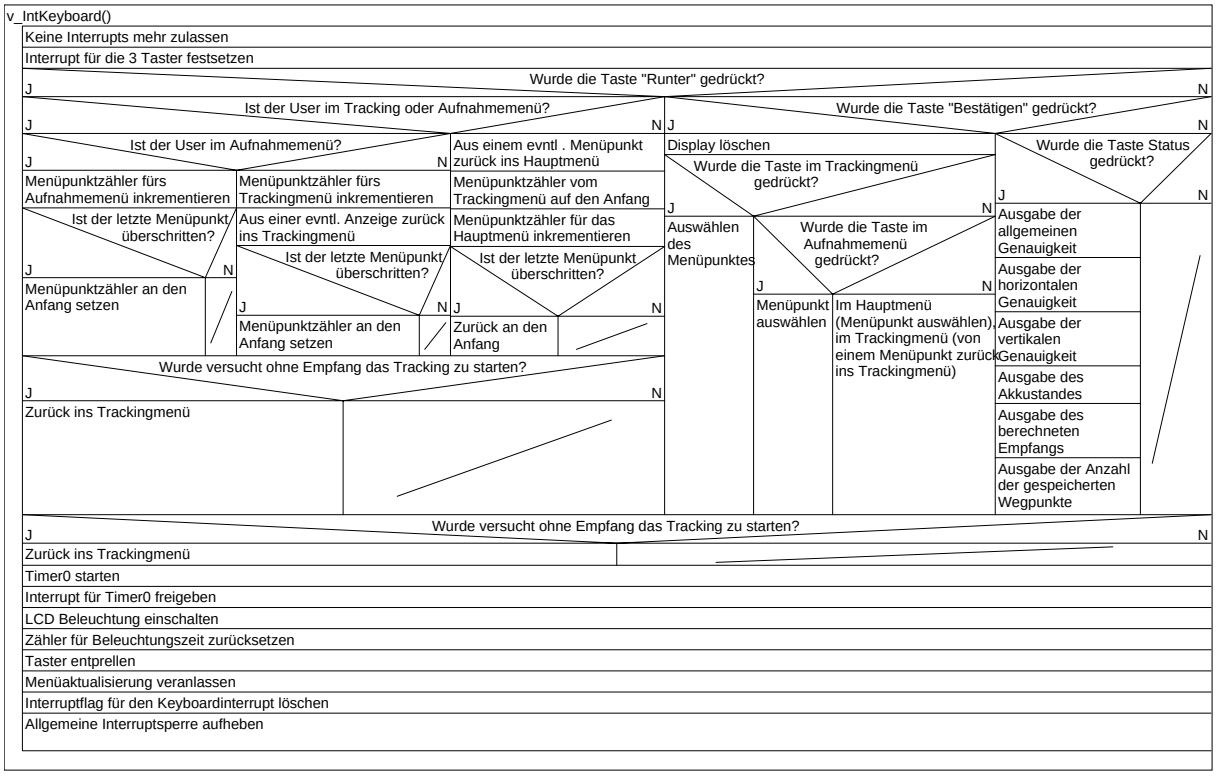
Struktogramm 3

Funktion: **v_Timer1Int**()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: In der ISR wird während einer Aufnahme alle 5s die aktuelle Koordinate als Wegpunkt im externen EEPROM abgespeichert. Alle 2s wird eine Menüaktualisierung veranlasst, der serielle Interrupt freigegeben und der Batteriestatus berechnet.
In einem Abstand von 30s wird hier der Empfang berechnet. Die Anzahl der gültigen Datensätze, welche in diesen 30s empfangen wurden, wird mit einem Richtwert verglichen. Ist dieser erreicht besteht ein Empfang von 100%, falls nicht, wird die prozentuale Annäherung dieses Richtwertes als Hinweis auf die Empfangsqualität ausgegeben.



Struktogramm 4

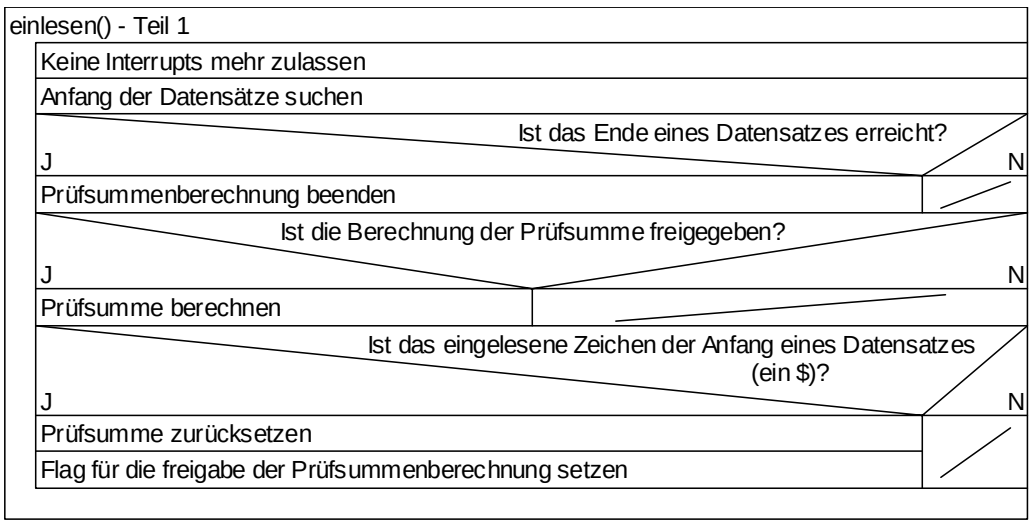
Funktion: **v_IntKeyboard**()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Bei jedem Tastendruck wird diese ISR abgearbeitet. Es wird analysiert, welche Taste auf welcher Menüebene gedrückt wurde. Nach jedem Tastendruck wird die Beleuchtung eingeschaltet und der Taster entprellt.



Struktogramm 5

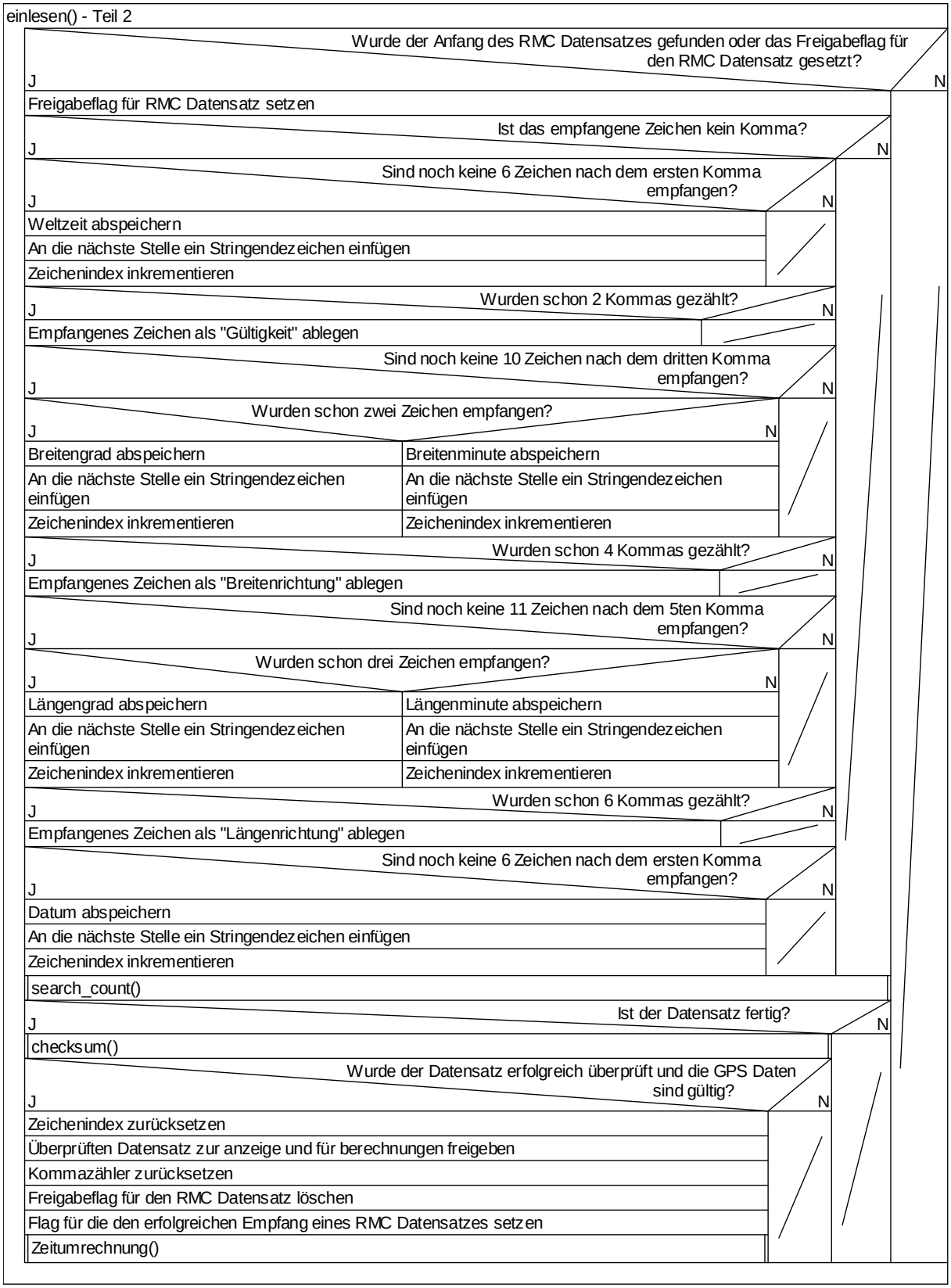
5.3.4 einlesen.c

Funktion: **einlesen**()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Die Funktion „einlesen()“ wird bei jedem seriellen Interrupt aufgerufen. Diese Funktion wurde zur besseren Darstellung in 5 Teile aufgegliedert.
Hier werden die ASCII Zeichen, welche vom GPS-Modul geliefert werden, analysiert und abgespeichert.
Über den Vergleich der letzten 6 empfangenen Zeichen mit einer vordefinierten Maske (z.B. „$GPRMC“) wird der Anfang eines Datensatzes gesucht.
Sobald der Anfang eines Datensatzes gefunden wurde, wird die Prüfsummenberechnung gestartet und jedes Zeichen mit den bereits empfangenen Zeichen logisch (EXOR) verknüpft.
Ist das Ende eines Datensatzes erreicht, (signalisiert durch das '*' Zeichen) wird die Berechnung der Prüfsumme gestoppt und über eine Funktion mit der überlieferten Prüfsumme verglichen. Stimmen beide Prüfsummen überein, wird der Datensatz für weitere Berechnungen freigegeben.
Für jeden Datensatz wurde ein Strukturvektor angelegt. D.h., es gibt 4 Strukturvektoren. Jeder dieser Vektoren beinhaltet 2 Strukturen, eine für die Empfangenen und eine für die überprüften Datensätze. Somit kann gewährleistet werden, dass die Berechnungen nur mit Daten durchgeführt werden, welche auf ihre Gültigkeit und Fehlerfreiheit überprüft wurden.
Die empfangenen Zeichen werden in einem Strukturvektor mit dem Index 0 gespeichert. Ist die Prüfsumme und die Gültigkeit der Daten übergeprüft und in Ordnung, werden diese in dem Strukturvektor unter dem Index 1 abgelegt.
Die Berechnung und die Anzeigen im Menü arbeiten nur mit den Daten, welche unter dem Index 1 gespeichert sind.



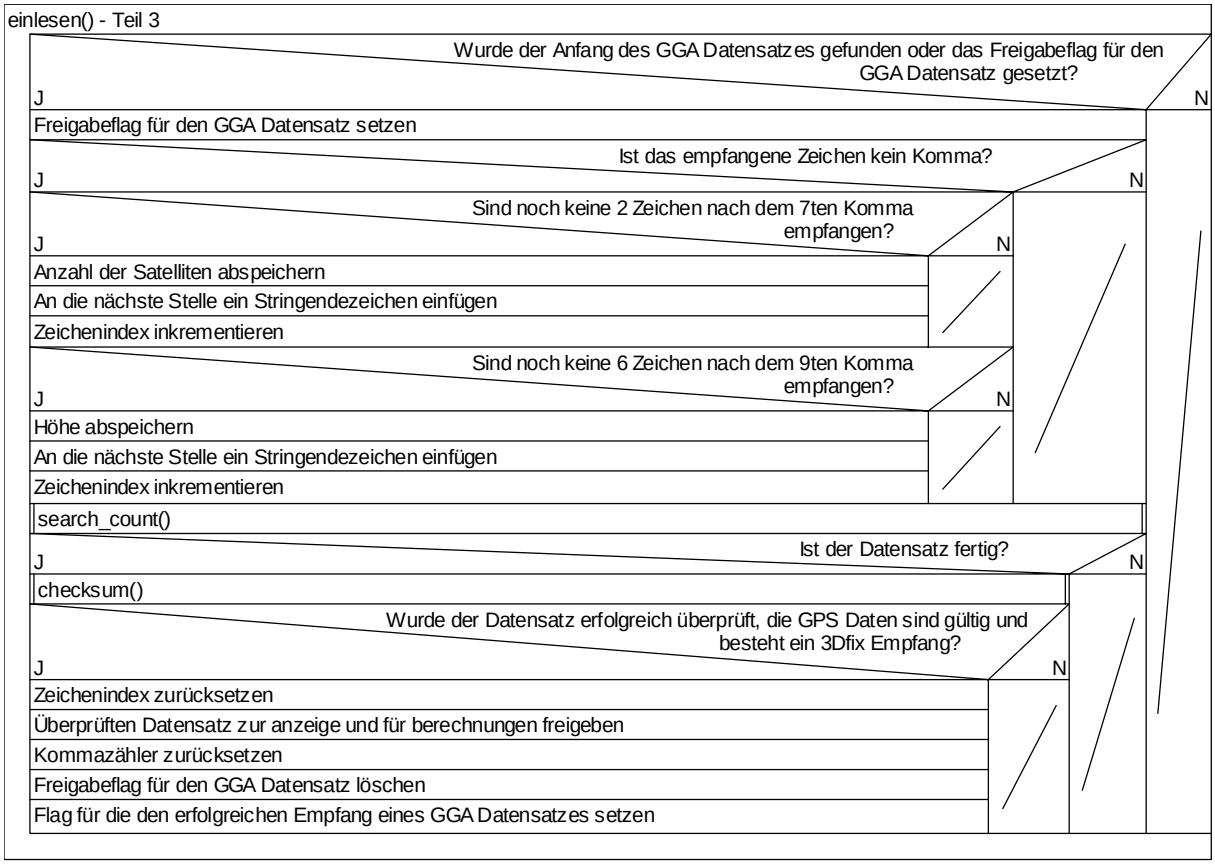
Struktogramm 6

Funktion: **einlesen**()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Nach dem der Anfang eines RMC Datensatzes gefunden wurde, wird ein Flag gesetzt, welches den Einlesevorgang des Datensatzes freigibt. Dadurch, dass die gesamte Funktion bei jedem Empfang eines einzelnen Zeichens ausgeführt wird, ist so gewährleistet das der angefangenen Datensatz fertig gestellt werden kann, bevor ein neuer Anfang gesucht wird.
Nach Abschluss des Datensatzes, oder wenn ein Fehler aufgetreten ist, wird das Flag gelöscht und ein neuer Anfang eines Datensatzes wird gesucht.
Daraufhin wird untersucht, ob das empfangene Zeichen ein Komma ist. Ist dies der Fall, wird eine extra Funktion gestartet, welche die Kommas zählt und kontrolliert, ob die maximale Anzahl der Kommas nicht überschritten wurde. (s. auch S. 32)
Ist das empfangene Zeichen kein Komma, wird je nach der Anzahl der bereits empfangenen Kommas und des Zeichenindex, das Zeichen an der entsprechende Stelle in der Struktur abgespeichert. Anschließend wird der Zeichenindex erhöht.
In den meisten Fällen werden die Daten als Strings abgelegt. Hier wird dann automatisch ein „Stringendezeichen“ an das Ende des Strings angehängt.
Für jeden Datensatz gibt es eine spezielle Filterung. Das heißt, je nach Datensatz (RMC, GGA, VTG oder GSA), wird ein anderer Filter angewendet. In einem solchen Filter werden die benötigten Daten aus den empfangenen Datensätzen entnommen und je nach Anzahl der empfangenen Kommas, an einer anderen Stelle, in der jeweiligen Struktur abgelegt.
Hier werden nur die Strukturen, der Strukturvektoren mit dem Index 0 verwendet.
Wenn der Datensatz fertig ist, wird die Prüfsumme verglichen. Der Datensatz wird auf die Gültigkeit der Daten überprüft und letztendlich wird der Datensatz für die Anzeige und die Berechnungen freigegeben. Die Daten werden durch das Kopieren unter einem neuen Index im selben Strukturvektor freigegeben.
Da in diesem Datensatz die Zeit übertragen wird, wird diese nach der Freigabe in der Funktion „zeitumrechnung()“ an die deutsche Zeitzone angepasst.



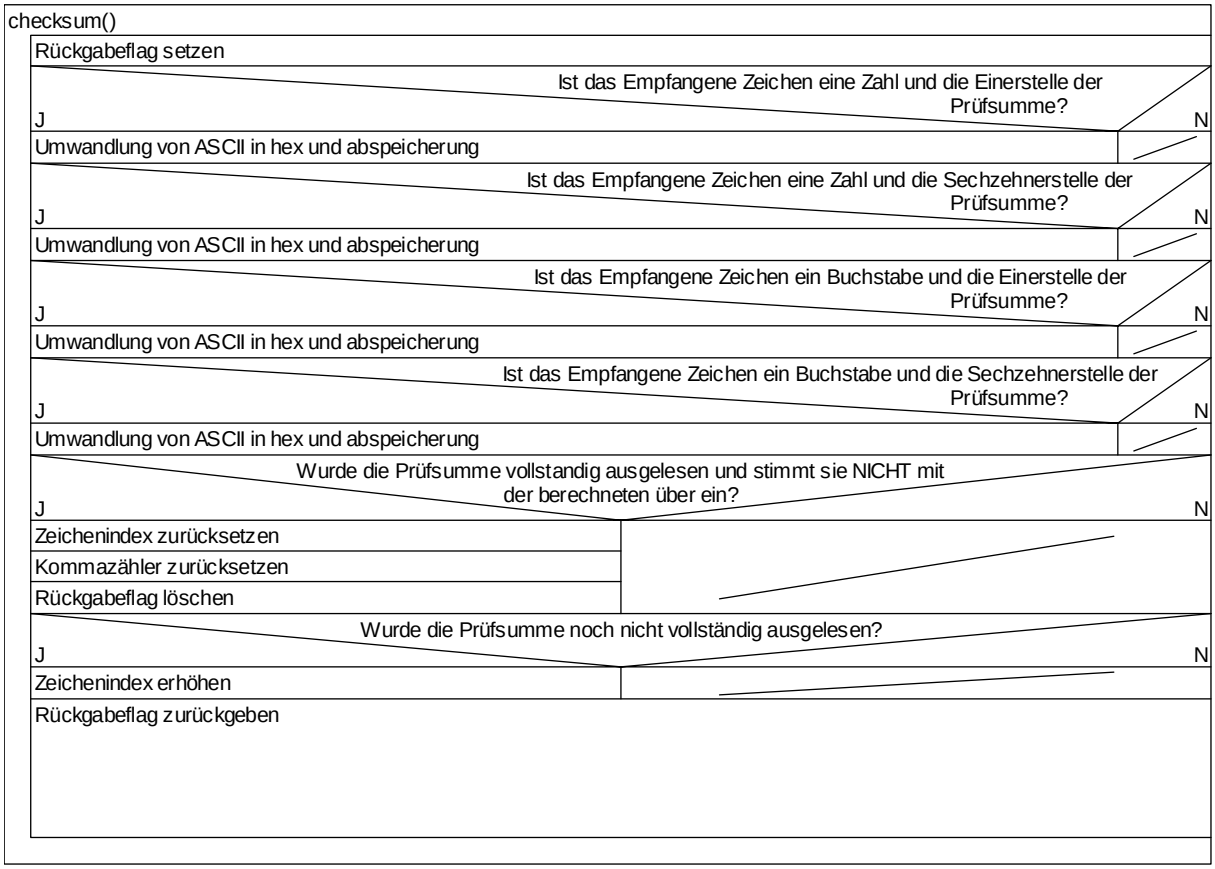
Struktogramm 7

Funktion: **einlesen**()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Der Ablauf, wie die Datensätze eingelesen werden ist bei allen Datensätzen gleich. Es werden lediglich andere Informationen aus den Datensätzen herausgefiltert (andere Filter angewendet).
Die Datensätze, welche die Höhe und die Geschwindigkeit enthalten, werden nur bei 3D-Fix Empfang freigegeben.
Da die anderen beiden Datensätze nach demselben Prinzip eingelesen werden, werden diese hier nicht weiter aufgeführt.


einlesen() - Teil 3
Wurde der Anfang des GGA Datensatzes gefunden oder das Freigabeflag für den GGA Datensatz gesetzt?
J
N
Freigabeflag für den GGA Datensatz setzen
Ist das empfangene Zeichen kein Komma?
J
N
Sind noch keine 2 Zeichen nach dem 7ten Komma empfangen?
J
N
Anzahl der Satelliten abspeichern
An die nächste Stelle ein Stringendezeichen einfügen
Zeichenindex inkrementieren
Sind noch keine 6 Zeichen nach dem 9ten Komma empfangen?
J
N
Höhe abspeichern
An die nächste Stelle ein Stringendezeichen einfügen
Zeichenindex inkrementieren
search_count()
Ist der Datensatz fertig?
J
N
checksum()
Wurde der Datensatz erfolgreich überprüft, die GPS Daten sind gültig und besteht ein 3Dfix Empfang?
J
N
Zeichenindex zurücksetzen
Überprüften Datensatz zur anzeige und für berechnungen freigeben
Kommazähler zurücksetzen
Freigabeflag für den GGA Datensatz löschen
Flag für die den erfolgreichen Empfang eines GGA Datensatzes setzen

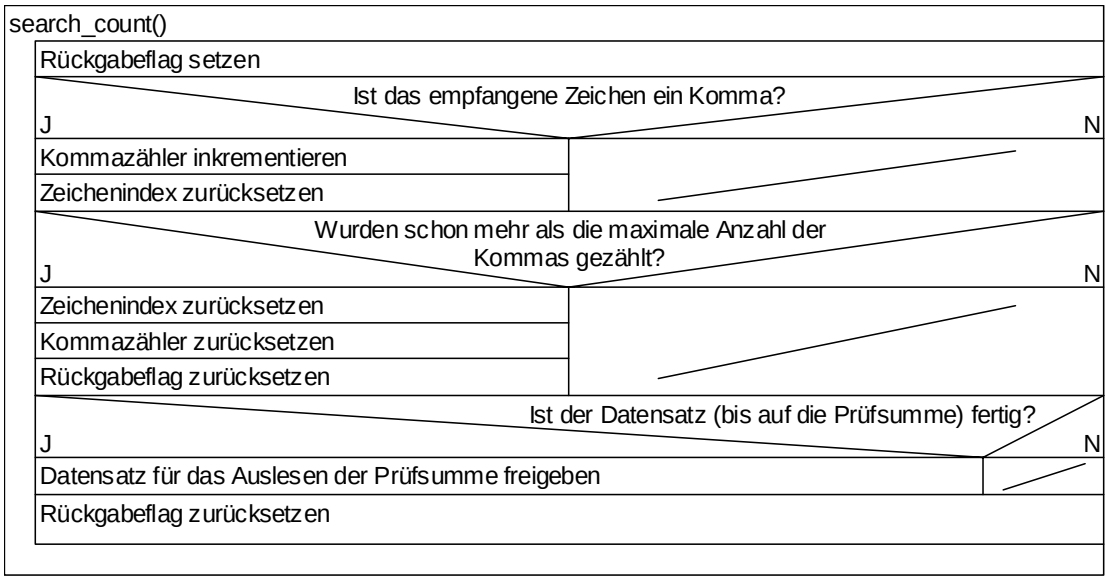
Struktogramm 8

Funktion: unsigned char **checksum**()
Übergabewerte: keine
Rückgabewert: Flag für die Erkennung fehlerhafter Datensätze
Beschreibung: Die Funktion „checksum()“ liest die vom Satelliten mit übertragene Prüfsumme aus und vergleicht diese mit der berechneten. Stimmt sie überein, ist alles in Ordnung. Stimmt sie nicht überein, wird ein Fehlerflag zurückgegeben. Dieses veranlasst, dass der fehlerhafte Datensatz nicht freigegeben wird.


checksum()
Rückgabeflag setzen
Ist das Empfangene Zeichen eine Zahl und die Einerstelle der Prüfsumme?
J
N
Umwandlung von ASCII in hex und abspeicherung
Ist das Empfangene Zeichen eine Zahl und die Sechzehnerstelle der Prüfsumme?
J
N
Umwandlung von ASCII in hex und abspeicherung
Ist das Empfangene Zeichen ein Buchstabe und die Einerstelle der Prüfsumme?
J
N
Umwandlung von ASCII in hex und abspeicherung
Ist das Empfangene Zeichen ein Buchstabe und die Sechzehnerstelle der Prüfsumme?
J
N
Umwandlung von ASCII in hex und abspeicherung
Wurde die Prüfsumme vollstandig ausgelesen und stimmt sie NICHT mit der berechneten über ein?
J
N
Zeichenindex zurücksetzen
Kommazähler zurücksetzen
Rückgabeflag löschen
Wurde die Prüfsumme noch nicht vollständig ausgelesen?
J
N
Zeichenindex erhöhen
Rückgabeflag zurückgeben

Struktogramm 9

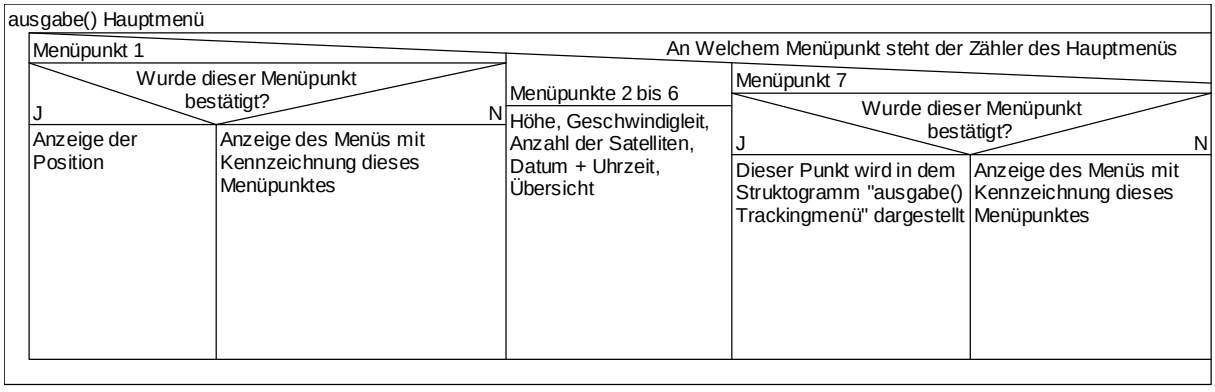
Funktion: unsigned char **search_count**(unsigned char)
Übergabewerte: Anzahl der Kommas, welche maximal in dem einzelnen Datensatz vorkommen dürfen.
Rückgabewert: Flag für die Erkennung fehlerhafter Datensätze
Beschreibung: Die Funktion „search_count()“ zählt die Kommas in einem Datensatz. Bei jedem empfangenen Komma wird der Kommazähler inkrementiert und der Zeichenindex zurückgesetzt.
Wurde die maximale Anzahl der erlaubten Kommas in diesem Datensatz überschritten, wird dieser gesperrt, d.h. nicht für die Prüfsummenkontrolle freigegeben und beim nächsten Empfang gleich wieder überschrieben.
Wenn das Ende des Datensatzes erreicht ist und alle Kommas ordnungsgemäß empfangen wurden, wird der Datensatz für den Prüfsummenvergleich freigegeben.



Struktogramm 10

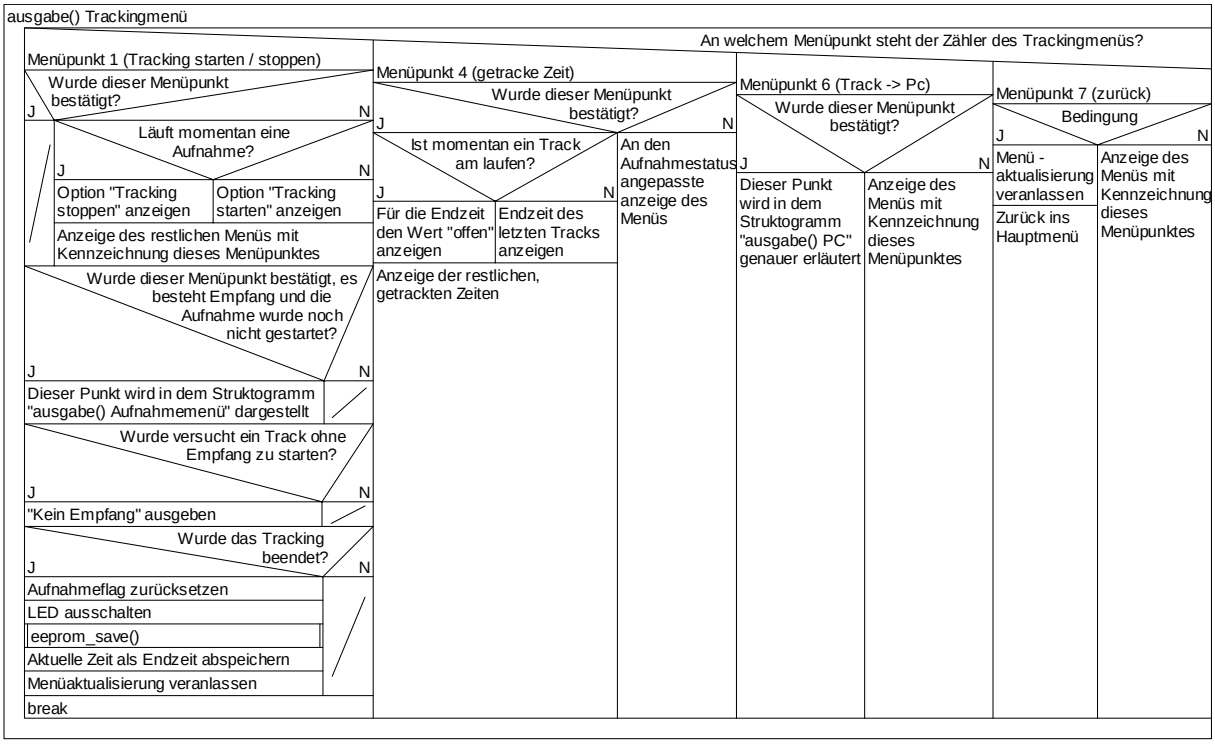
5.3.5 ausgabe.c

Funktion: **ausgabe**()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Alle Ausgabemenüs sind über jeweils einen Index für die Menüebene und einen Menüpunktzähler realisiert. Bei Betätigung der Taste „↓“ wird der Menüpunktzähler um eins erhöht und bei Betätigung der Taste „Enter“, die Menüebene. Die Funktion „ausgabe()“ wertet diese dann aus.
Über den Menüpunktzähler springt das Programm an den jeweiligen Menüpunkt und über die Menüebene wird nun entweder der Inhalt des Menüpunktes oder das Menü mit markiertem Menüpunkt angezeigt.
Die Menüpunkte 2 bis 6 sind wie der Menüpunkt 1 aufgebaut.
Aufgrund von Platzgründen wurde die Darstellung des Struktogrammes in die verschiedenen Menüs aufgeteilt. Es wird zwischen dem Hauptmenü, dem Trackingmenü und dem Aufnahmemenü unterschieden.
Das Hauptmenü besteht aus 7 Menüpunkten. Hier sind die alle Daten abrufbar die nicht mit einem „Tracking“ in Verbindung stehen. Das sind z.B. die aktuelle Position, die Höhe, die Uhrzeit, usw..
Nach einer Unterbrechung der Spannungsversorgung werden als Platzhalter für die Daten, „x“e angezeigt. Erst bei korrektem Empfang werden diese durch die Daten des Satelliten ersetzt.
Wird jedoch der Empfang unterbrochen, bleiben die zuletzt empfangenen Daten stehen.


ausgabe() Hauptmenü
An Welchem Menüpunkt steht der Zähler des Hauptmenüs
Menüpunkt 1
Wurde dieser Menüpunkt bestätigt?
J
N
Anzeige der Position
Anzeige des Menüs mit Kennzeichnung dieses Menüpunktes
Menüpunkte 2 bis 6
Höhe, Geschwindigleit, Anzahl der Satelliten, Datum + Uhrzeit, Übersicht
Menüpunkt 7
Wurde dieser Menüpunkt bestätigt?
J
N
Dieser Punkt wird in dem Struktogramm "ausgabe() Trackingmenü" dargestellt
Anzeige des Menüs mit Kennzeichnung dieses Menüpunktes

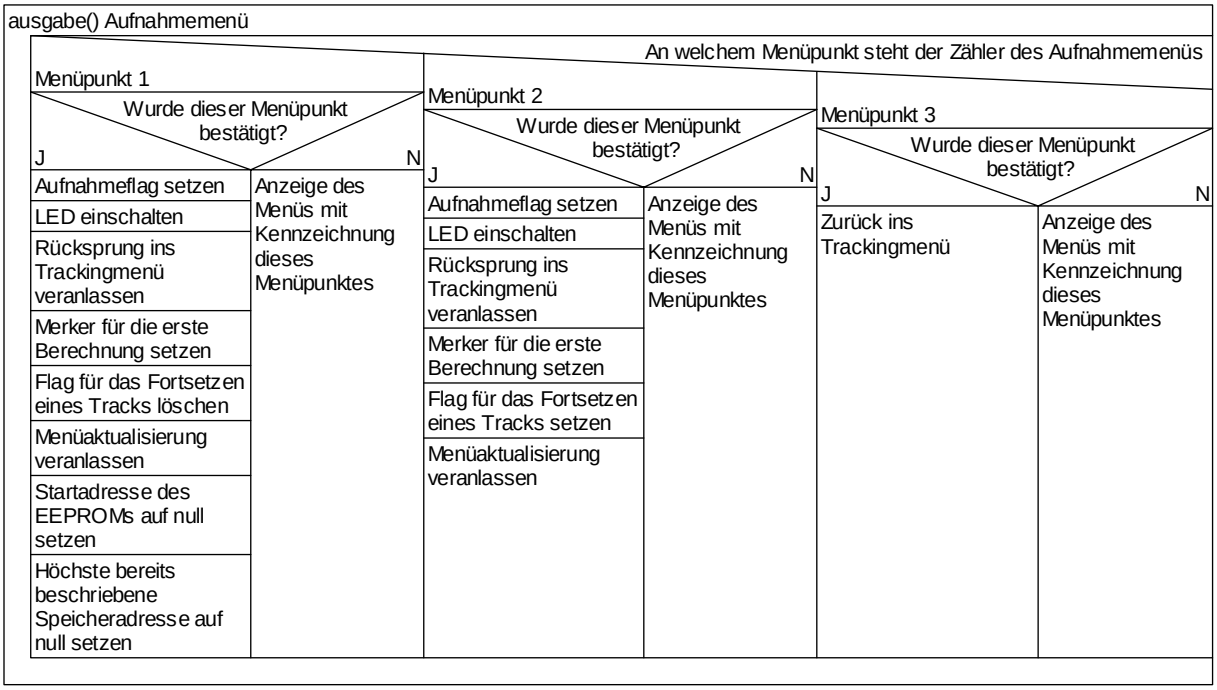
Struktogramm 11

Funktion: ausgabe()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Über das Trackingmenü werden alle Informationen bezüglich der Aufnahme zugänglich gemacht. Dies sind unter anderem die Aufnahmezeit oder die zurückgelegte Strecke.
Bei der Anzeige der Endzeit wird bei einem aktiven Tracking keine Zeit sondern „offen“ angezeigt.
Über den ersten Menüpunkt kann das Tracking gestartet und auch wieder gestoppt werden. Je nach dem, ob momentan ein Tracking aktiv ist, wird die Option „Tracking stoppen“ oder „Tracking starten“, angeboten.
Um ein Tracking zu starten muss Empfang bestehen, ist das nicht der Fall, wird auf dem Display „kein Empfang!“ dargestellt. Besteht Empfang, gelangt der Benutzer in das Aufnahmemenü, welches im folgenden Abschnitt genauer erläutert wird.
Über das Trackingmenü kann auch die Datenübertragung zum PC durchgeführt werden. Dieser Punkt wird an anderer Stelle verdeutlicht.
Alle Informationen, welche hier abrufbar sind, werden im internen EEPROM gespeichert. D.h., durch das Ausschalten des Gerätes, gehen diese nicht verloren.
Die Menüpunkte 2, 3 und 5 sind an sich wie der Menüpunkt 6 aufgebaut. Mit dem Unterschied das wenn der Menüpunkt bestätigt wurde, der Inhalt des Menüpunktes angezeigt wird.



Struktogramm 12

Funktion: ausgabe()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: In dem Aufnahmemenü kann ausgewählt werden, ob ein neuer Track gestartet, oder ein angefangener Track fortgesetzt werden soll. Der Unterschied besteht in dem setzten eines Flags und dem Rücksetzen der Speicheradressen des externen EEPROMs.



Struktogramm 13

Funktion: ausgabe() - Trackingmenü
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Hier wird der komplette Speicherinhalt über die USB Schnittstelle übertragen. Das Ende der Übertragung wird mit dem Line-Feed Zeichen signalisiert.
Eine solche Datenübertragung sieht dann folgendermaßen aus:

4741.6480N00906.8952E4741.7165N00906.8623E4741.7519N00906.8390E…

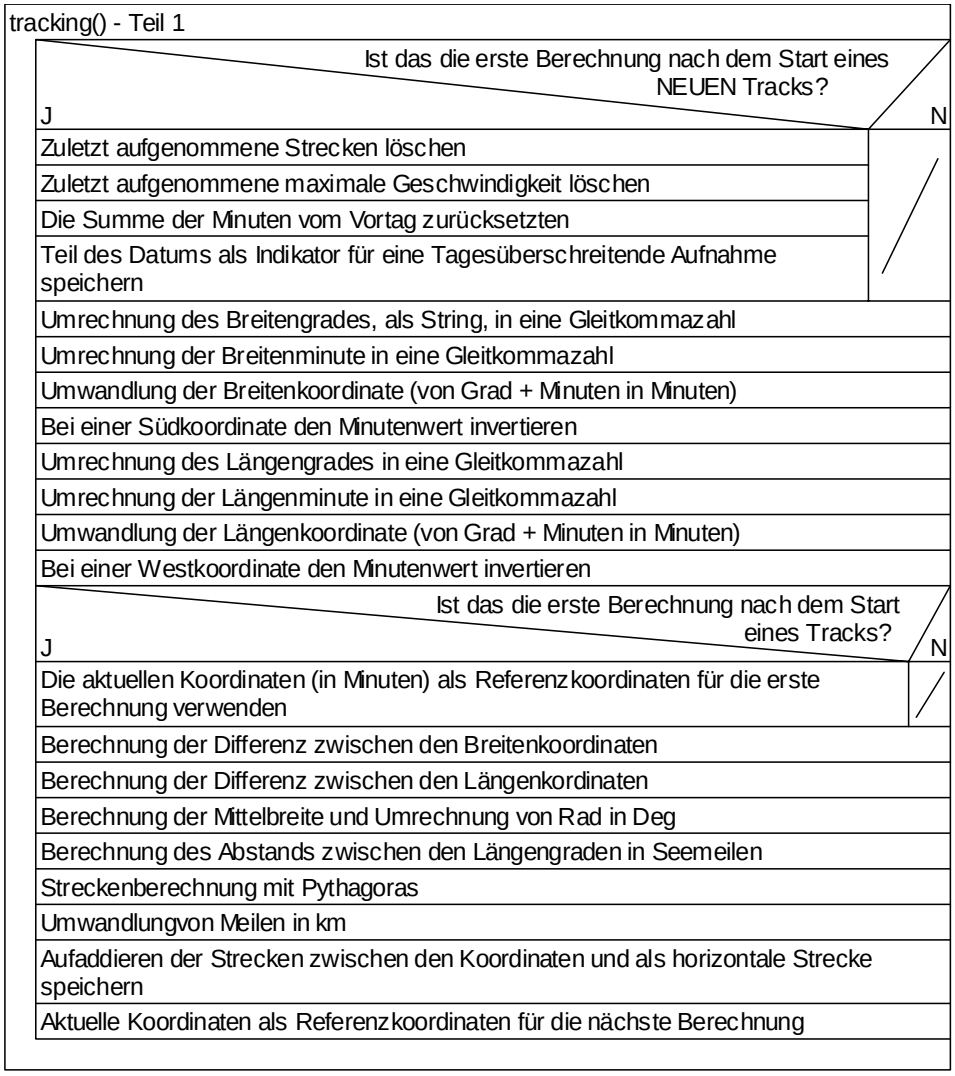
4741.6480	Breitenkoordinate
N	Breitenrichtung
00906.8952	Längenkoordinate
E	Längenrichtung

ausgabe() Track -> PC	
Wurde dieser Menüpunkt bestätigt? J	N
Benutzerführung	Anzeige des Menüs mit Kennzeichnung dieses Menüpunktes
Adresse des EEPROMS auf null setzen	
Bis alle Wegpunkte übertragen wurden und mindestens ein Wegpunkt abgespeichert ist	
Wegpunkt aus dem Speicher lesen	
Wegpunkt über die serielle Schnittstelle übertragen	
Als Kennzeichnung des Endes, ein Linefeed senden	
Benutzerführung	
Zurück ins Trackingmenü	

Struktogramm 14

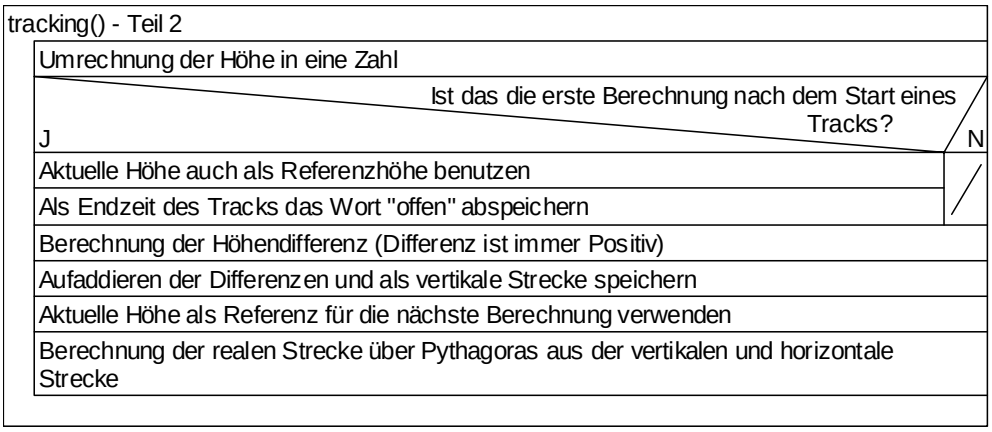
5.3.6 tracking.c

Funktion: tracking()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Die Funktion „tracking()“ beinhaltet alle Berechnungen die während der Aufnahme vorgenommen werden.
Zur besseren Darstellung wurde diese Funktion in mehrere Teile aufgegliedert.
In Teil 1 wird die Streckenberechnung vorgenommen.


tracking() - Teil 1
Ist das die erste Berechnung nach dem Start eines NEUEN Tracks?
J
N
Zuletzt aufgenommene Strecken löschen
Zuletzt aufgenommene maximale Geschwindigkeit löschen
Die Summe der Minuten vom Vortag zurücksetzten
Teil des Datums als Indikator für eine Tagesüberschreitende Aufnahme speichern
Umrechnung des Breitengrades, als String, in eine Gleitkommazahl
Umrechnung der Breitenminute in eine Gleitkommazahl
Umwandlung der Breitenkoordinate (von Grad + Minuten in Minuten)
Bei einer Südkoordinate den Minutenwert invertieren
Umrechnung des Längengrades in eine Gleitkommazahl
Umrechnung der Längenminute in eine Gleitkommazahl
Umwandlung der Längenkoordinate (von Grad + Minuten in Minuten)
Bei einer Westkoordinate den Minutenwert invertieren
Ist das die erste Berechnung nach dem Start eines Tracks?
J
N
Die aktuellen Koordinaten (in Minuten) als Referenzkoordinaten für die erste Berechnung verwenden
Berechnung der Differenz zwischen den Breitenkoordinaten
Berechnung der Differenz zwischen den Längenkordinaten
Berechnung der Mittelbreite und Umrechnung von Rad in Deg
Berechnung des Abstands zwischen den Längengraden in Seemeilen
Streckenberechnung mit Pythagoras
Umwandlungvon Meilen in km
Aufaddieren der Strecken zwischen den Koordinaten und als horizontale Strecke speichern
Aktuelle Koordinaten als Referenzkoordinaten für die nächste Berechnung

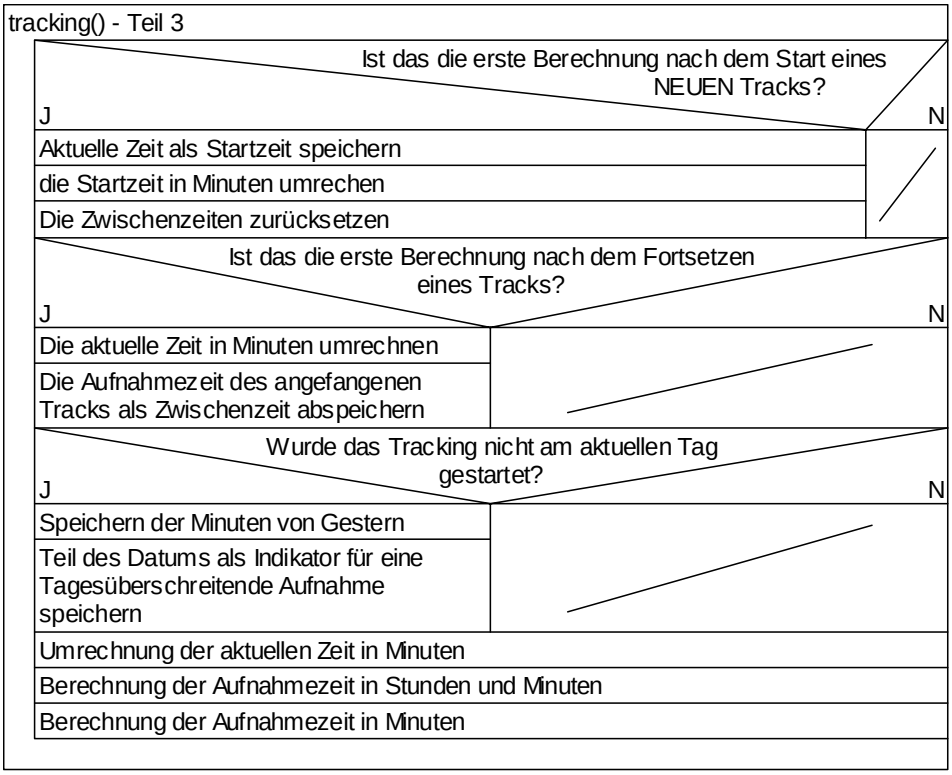
Struktogramm 15

Funktion: tracking()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Das folgende Struktogramm schildert die Berechnungen der vertikalen Strecke und der realen Strecke.
Da die Höhe, wie alle Daten von dem GPS-Modul, in ASCII Zeichen übertragen wird, muss diese in eine Zahl umgewandelt werden.
Die vertikale Strecke wird aus der Summe der Differenzen zwischen der aktuellen Höhe und der Höhe der letzten Position berechnet.
Zur Berechnung der realen Strecke wird über den Satz des Pythagoras die horizontale Strecke mit der vertikalen Strecke verrechnet.


tracking() - Teil 2
Umrechnung der Höhe in eine Zahl
Ist das die erste Berechnung nach dem Start eines Tracks?
J
N
Aktuelle Höhe auch als Referenzhöhe benutzen
Als Endzeit des Tracks das Wort "offen" abspeichern
Berechnung der Höhendifferenz (Differenz ist immer Positiv)
Aufaddieren der Differenzen und als vertikale Strecke speichern
Aktuelle Höhe als Referenz für die nächste Berechnung verwenden
Berechnung der realen Strecke über Pythagoras aus der vertikalen und horizontale Strecke

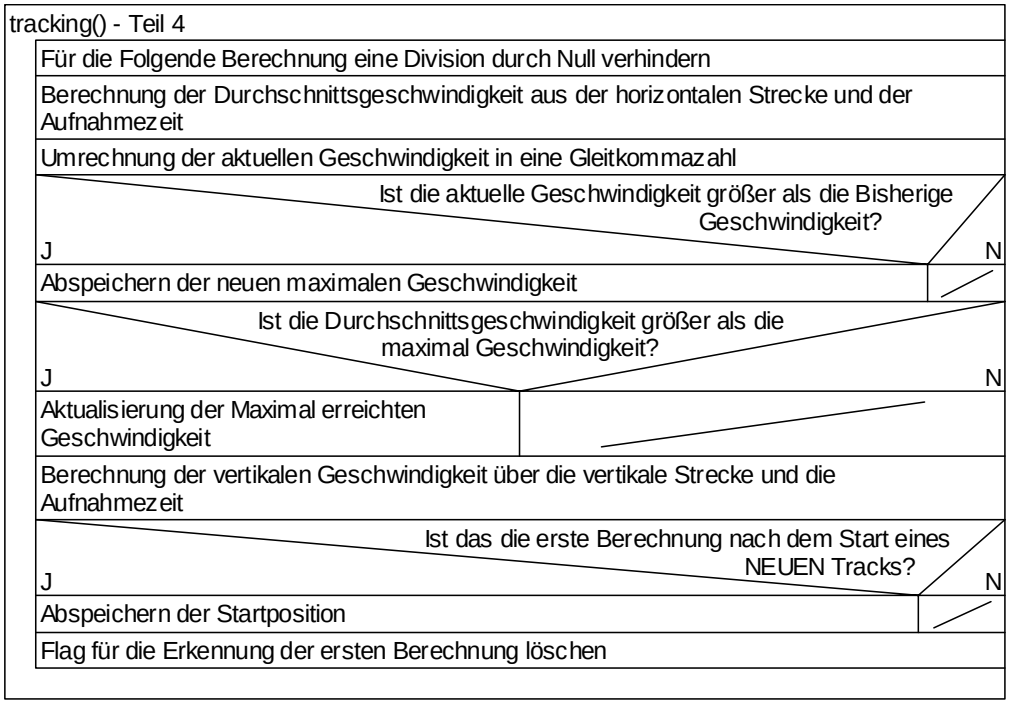
Struktogramm 16

Funktion: tracking()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Um die Aufnahmezeit berechnen zu können muss die Startzeit und die aktuelle Zeit in Minuten umgerechnet werden. Die Differenz aus diesen beiden Zeiten ergibt die Aufnahmezeit.
Problematisch wird es nur, wenn die Aufnahme über einen Tag hinausgeht. Z.B. wenn von 23 Uhr bis 1 Uhr morgens getrackt wird. Um das Problem zu lösen muss erkannt werden, wann ein neuer Tag beginnt. Ist das der Fall wird die verstrichene Zeit vom Vortag, in dem Beispiel 60min, abgespeichert und zu der Differenz zwischen 0 Uhr und der aktuellen Zeit aufaddiert.


tracking() - Teil 3
Ist das die erste Berechnung nach dem Start eines NEUEN Tracks?
J
N
Aktuelle Zeit als Startzeit speichern
die Startzeit in Minuten umrechen
Die Zwischenzeiten zurücksetzen
Ist das die erste Berechnung nach dem Fortsetzen eines Tracks?
J
N
Die aktuelle Zeit in Minuten umrechnen
Die Aufnahmezeit des angefangenen Tracks als Zwischenzeit abspeichern
Wurde das Tracking nicht am aktuellen Tag gestartet?
J
N
Speichern der Minuten von Gestern
Teil des Datums als Indikator für eine Tagesüberschreitende Aufnahme speichern
Umrechnung der aktuellen Zeit in Minuten
Berechnung der Aufnahmezeit in Stunden und Minuten
Berechnung der Aufnahmezeit in Minuten

Struktogramm 17

Funktion: tracking()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Die Durchschnittsgeschwindigkeit wird über die Aufnahmezeit und die horizontale Strecke berechnet.
Um die maximal erreichte Geschwindigkeit zu ermitteln wird die aktuelle Geschwindigkeit mit dem bisher erreichten Höchstwert verglichen.
Da die Geschwindigkeit vom Satelliten übertragen, und die Durchschnittsgeschwindigkeit über die Zeit und die Strecke berechnet wird, kann es vorkommen das die Durchschnittsgeschwindigkeit größer ist als die höchste, vom Satelliten übertragene Geschwindigkeit. Aufgrund dessen, wird auch die Durchschnittsgeschwindigkeit mit der bisher ermittelten Maximalgeschwindigkeit verglichen und eventuell angepasst.
Beim ersten Durchlauf der Berechnungen wird die Position, bestehend aus Koordinate und Höhe, als Startposition abgespeichert.

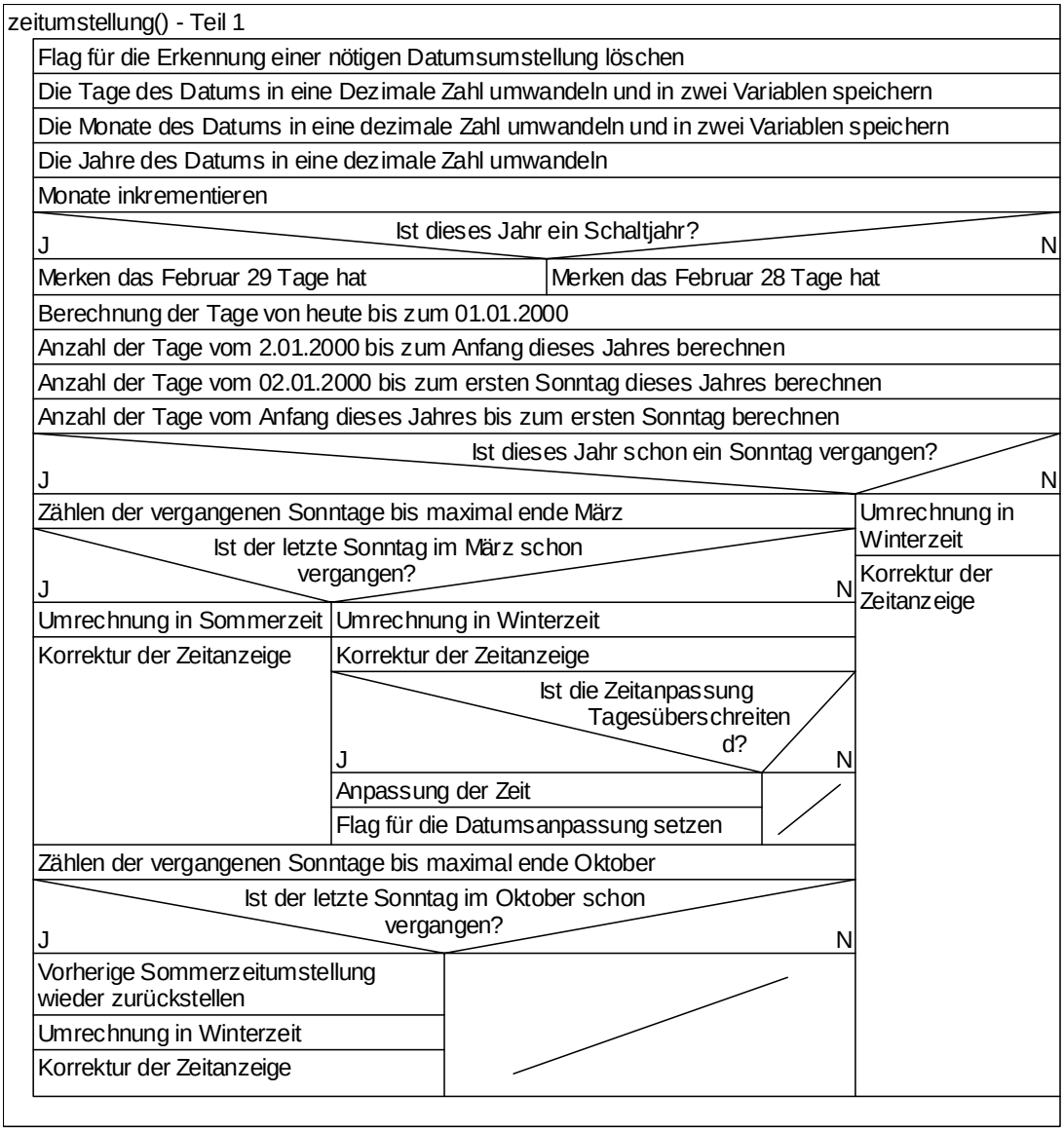

tracking() - Teil 4
Für die Folgende Berechnung eine Division durch Null verhindern
Berechnung der Durchschnittsgeschwindigkeit aus der horizontalen Strecke und der Aufnahmezeit
Umrechnung der aktuellen Geschwindigkeit in eine Gleitkommazahl
Ist die aktuelle Geschwindigkeit größer als die Bisherige Geschwindigkeit?
J
N
Abspeichern der neuen maximalen Geschwindigkeit
Ist die Durchschnittsgeschwindigkeit größer als die maximal Geschwindigkeit?
J
N
Aktualisierung der Maximal erreichten Geschwindigkeit
Berechnung der vertikalen Geschwindigkeit über die vertikale Strecke und die Aufnahmezeit
Ist das die erste Berechnung nach dem Start eines NEUEN Tracks?
J
N
Abspeichern der Startposition
Flag für die Erkennung der ersten Berechnung löschen

Struktogramm 18

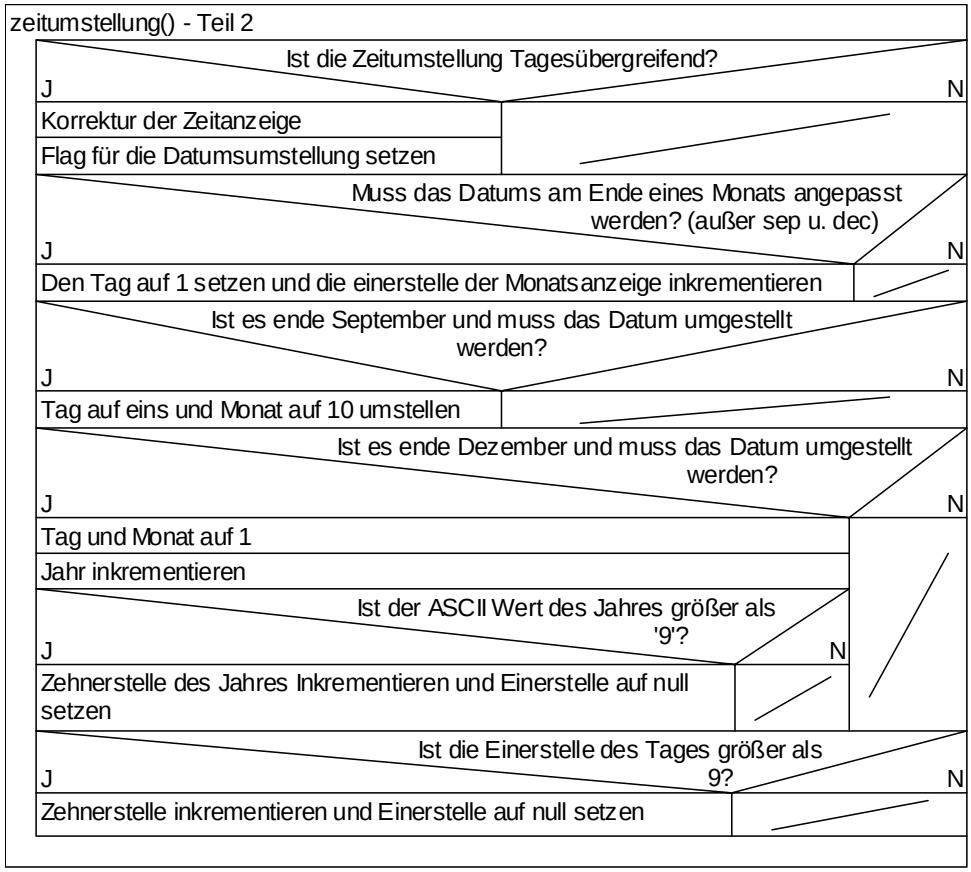
5.3.7 zeitumstellung.c

Funktion: zeitumstellung()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Die automatische Zeitanpassung an die deutsche Sommer- und Winterzeit ist recht aufwendig und komplex.
Über das GPS-Modul werden die Weltzeit und das Datum übertragen.
Aus diesen Informationen muss zunächst ermittelt werden was für ein Wochentag heute ist. Dazu wird zusätzlich ein Referenztag, der ein Sonntag ist, benötigt. Da nur die Information relevant ist, ob heute ein Sonntag ist, werden die Tage von heute bis zu diesem Referenztag unter Berücksichtigung der Schaltjahre gezählt. Ist die Anzahl der Tage durch 7 teilbar, ist der heutige Tag ein Sonntag.
Jetzt muss noch ermittelt werden, ob schon der letzte Sonntag im März oder der letzte Sonntag im Oktober verstrichen ist. Je nach dem besteht Sommer- oder Winterzeit.
Wenn das bekannt ist, ist es mit dem Aufaddieren von einer oder zwei Stunden aber noch nicht getan. Wenn z.B. 2 Stunden aufaddiert werden, muss verhindert werden, dass eine Zeit von z.B. 25 Uhr angezeigt wird. Und wenn wie in dem Beispiel die Zeitumstellung tagesübergreifend ist muss zudem auch das Datum angepasst werden.
Die Tatsache, dass die Daten als ASCII Zeichen abgespeichert

werden, macht die Sache zusätzlich aufwendiger. Somit geht die einfache Rechnung „9“ + 2 = „11“ leider nicht auf. Stattdessen ist „9“ + 2 = „;“.
Es wäre zwar möglich die ASCII Zeichen in Zahlen umzuwandeln, doch das würde mehr Speicher benötigen.


zeitumstellung() - Teil 1
Flag für die Erkennung einer nötigen Datumsumstellung löschen
Die Tage des Datums in eine Dezimale Zahl umwandeln und in zwei Variablen speichern
Die Monate des Datums in eine dezimale Zahl umwandeln und in zwei Variablen speichern
Die Jahre des Datums in eine dezimale Zahl umwandeln
Monate inkrementieren
Ist dieses Jahr ein Schaltjahr?
J
N
Merken das Februar 29 Tage hat
Merken das Februar 28 Tage hat
Berechnung der Tage von heute bis zum 01.01.2000
Anzahl der Tage vom 2.01.2000 bis zum Anfang dieses Jahres berechnen
Anzahl der Tage vom 02.01.2000 bis zum ersten Sonntag dieses Jahres berechnen
Anzahl der Tage vom Anfang dieses Jahres bis zum ersten Sonntag berechnen
Ist dieses Jahr schon ein Sonntag vergangen?
J
N
Zählen der vergangenen Sonntage bis maximal ende März
Umrechnung in Winterzeit
Ist der letzte Sonntag im März schon vergangen?
J
N
Korrektur der Zeitanzeige
Umrechnung in Sommerzeit
Umrechnung in Winterzeit
Korrektur der Zeitanzeige
Korrektur der Zeitanzeige
Ist die Zeitanpassung Tagesüberschreiten d?
J
N
Anpassung der Zeit
Flag für die Datumsanpassung setzen
Zählen der vergangenen Sonntage bis maximal ende Oktober
Ist der letzte Sonntag im Oktober schon vergangen?
J
N
Vorherige Sommerzeitumstellung wieder zurückstellen
Umrechnung in Winterzeit
Korrektur der Zeitanzeige

Struktogramm 19


zeitumstellung() - Teil 2
Ist die Zeitumstellung Tagesübergreifend?
J
N
Korrektur der Zeitanzeige
Flag für die Datumsumstellung setzen
Muss das Datums am Ende eines Monats angepasst werden? (außer sep u. dec)
J
N
Den Tag auf 1 setzen und die einerstelle der Monatsanzeige inkrementieren
Ist es ende September und muss das Datum umgestellt werden?
J
N
Tag auf eins und Monat auf 10 umstellen
Ist es ende Dezember und muss das Datum umgestellt werden?
J
N
Tag und Monat auf 1
Jahr inkrementieren
Ist der ASCII Wert des Jahres größer als '9'?
J
N
Zehnerstelle des Jahres Inkrementieren und Einerstelle auf null setzen
Ist die Einerstelle des Tages größer als 9?
J
N
Zehnerstelle inkrementieren und Einerstelle auf null setzen

Struktogramm 20

5.3.8 funktionen.c

Die 3 Funktionen copy(), comp() und potenz() wurden geschrieben um Speicherplatz einzusparen. Die Funktionen gibt es auch, in abgewandelter Art, in den Bibliotheken string.h und math.h.

Funktion: void **copy**(char *, char *)
Übergabewerte: 1. Die Adresse des Zielstrings
2. Die Adresse des Quellstrings.
Rückgabewert: keiner
Beschreibung: Ermöglicht das Kopieren eines Strings in einen anderen.

Funktion: char **comp**(char *, char *)
Übergabewerte: 1. Startadresse des ersten Strings
2. Startadresse des zweiten Strings
Rückgabewert: Bei Gleichheit wird eine 1, und bei Ungleichheit wird eine 0 zurückgegeben.
Beschreibung: Ermöglicht das Vergleichen zweier Strings.

Funktion: double **potenz**(double, char)
Übergabewerte: 1. Die Basis
2. Den Exponent
Rückgabewert: Das Ergebnis
Beschreibung: Ermöglicht das Potenzieren eines Wertes.

5.3.9 eeprom.c

Funktion: eeprom_save()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Speichert alle Daten, die im Trackingmenü abgerufen werden können und die letzte Speicheradresse des externen EEPROMs, im internen EEPROM.

Funktion: eeprom_load()
Übergabewerte: keine
Rückgabewert: keiner
Beschreibung: Lädt die Daten aus dem internen EEPROM.

Funktion: void **char_write**(unsigned char, unsigned char*, unsigned char*)
Übergabewerte: 1. Anzahl der Zeichen
2. Zeiger auf die Adresse des internen EEPROMs
3. Zeiger auf die Startadresse der Zeichenkette
Rückgabewert: keiner
Beschreibung: Speichert Zeichenketten im internen EEPROM

Funktion: void **char_read**(unsigned char, unsigned char*, unsigned char*)
Übergabewerte: 1. Anzahl der Zeichen
2. Zeiger auf die Adresse des internen EEPROMs
3. Zeiger auf die Startadresse der Zeichenkette
Rückgabewert: keiner
Beschreibung: Lädt Zeichenketten aus dem internen EEPROM

Funktion: void **float_write**(unsigned char*, float*)
Übergabewerte: 1. Zeiger auf die Adresse des internen EEPROMs
2. Zeiger auf die Adresse der Float-Variable
Rückgabewert: keiner
Beschreibung: Speichert Float-Variablen in das internen EEPROM

Funktion: void **float_read**(unsigned char*, float*)
Übergabewerte: 1. Zeiger auf die Adresse des internen EEPROMs
2. Zeiger auf die Adresse der Float-Variable
Rückgabewert: keiner
Beschreibung: Lädt Float-Variablen aus dem internen EEPROM

5.4 Software (PC)

5.4.1 Allgemeines:

Die Software wurde mit dem Visual Studio 2005 entwickelt. Sie ist komplett in der Objektorientierten Sprache Visual Basic 2005 geschrieben.
Das Programm ist 252KByte groß, wobei das Bild von dem Wilkommensfenster und das Bild für das Icon inbegriffen sind. Ohne sind es etwa 60KByte

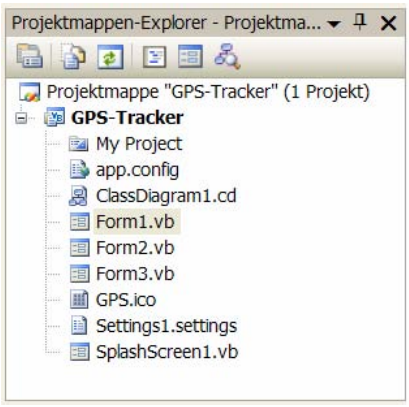


Bild 12

Das Programm besteht aus 9 Dateien. Davon sind 3 Dateien Programmfenster (Form1_Main.vb, Form2_Info.vb, Form3_Einstellungen.vb), 1 Datei ist das Icon (GPS.ico), 2 Dateien verwalten die Einstellungen (app.config, settings1.settings) und 1 Datei ist das Wilkommensfenster (SplashScreen1.vb).
Die Datei Form1_Main.vb ist die wichtigste Datei. Über diese Datei wird der komplette Ablauf des Programms gesteuert.

Kurzer Exkurs

Die Try - Catch Anweisung bietet eine Möglichkeit, einige oder alle möglichen Fehler zu behandeln, die in einem bestimmten Codeblock auftreten können, während dieser Code noch ausgeführt wird.
Entsteht ein Fehler in einem Try Block wird der restliche Code in diesem Block übersprungen und der Code im Catch Block wird abgearbeitet. Entsteht kein Fehler wird der Code im Catch Block nicht ausgeführt.

5.4.2 Klassen Übersicht

Settings
Ermöglicht das Speichern Benutzerspezifischer Daten. Über diese Klasse können Einstellungen gespeichert werden.
Sie wird für das Speichern des Portnamens, des Speicherpfades und des Hauptpfades verwendet.

SerialPort
Die Klasse SerialPort stellt eine Anschlussresaurce dar. Über diese Klasse können die Schnittstellen verwaltet werden.
Hier wird sie zur Verwaltung des USB Ports verwendet.
Diese Klasse ist neu in .Net Framework v2.0. Aufgrund dessen sollte das Programm nicht mit früheren Framework Versionen, oder der Version 3.5 ausgeführt werden.

SaveFileDialog
Fordert den Benutzer zur Auswahl eines Speicherorts für eine Datei auf.

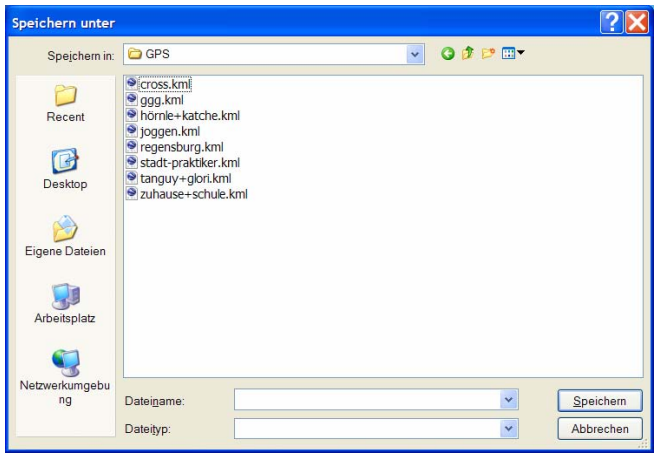


Bild 13

Über diese Klasse kann das Fenster Verwaltet werden. Das Fenster wird angezeigt, wenn ein Track vom Handgerät aus, übertragen wurde. Der Benutzer kann ein Verzeichnis auswählen, in dem der Track gespeichert werden soll

FolderBrowserDialog
Fordert den Benutzer auf, einen Ordner auszuwählen.
Wird ähnlich wie der SaveFileDialog verwaltet.

Bild 14

Wenn das Programm unter einem neuen Pfad zum ersten Mal ausgeführt wird, wird der Benutzer aufgefordert über dieses Fenster einen Speicherpfad für die künftigen Tracks auszuwählen. Über diesen Speicherpfad wird zudem die Liste aktualisiert.

Form
Stellt ein Fenster oder ein Dialogfeld dar, das die Benutzeroberfläche einer Anwendung bildet.
In der GPS-Tracker Software sind 3 dieser Klassen vorhanden.

1. Form1_Main stellt das Hauptprogramm dar
2. Form2_Info ist ein Fenster mit einem Informationstext zu der Software
3. Form3_Einstellungen beinhaltet die Verwaltung der Einstellungen

Timer
Erstellt wiederkehrende Ereignisse in einer Anwendung.
Wird lediglich zur Kommunikation zwischen den Threads verwendet.

StreamWriter
Ermöglicht das Schreiben von Zeichen in einer Datei. Zudem können über diese Klasse auch Dateien erstellt werden.
Mit dieser Klasse werden die KML-Dateien erzeugt und beschrieben.

StreamReader
Ist das Gegenstück von der Klasse StreamWriter. Sie ermöglicht das Auslesen einer Datei.
Über den StreamReader, werden beim anklicken einer Datei aus der Liste, die Dateien nach bestimmten Schlüsselwörtern durchsucht.

DirectoryInfo
Ermöglicht das Erstellen, Verschieben und Auflisten von Verzeichnissen und Unterverzeichnissen.
Wird benötigt um die Liste der Tracks zu aktualisieren und zu Verwalten.

FileInfo
Stellt Methoden zum Erstellen, Kopieren, Löschen, Verschieben und Öffnen von Dateien bereit.
Wird ebenfalls zur Verwaltung der Liste benötigt

Textbox
Stellt ein textbasiertes Steuerelement bereit, das dem Benutzer die Eingabe von Text ermöglicht.
Wird zur Kommunikation zwischen Threads und zur Anzeige des Speicherpfades in Form1_Einstellungen beansprucht.

Listbox
Stellt ein Listenfeld-Steuerelement dar, das die Auswahl eines oder mehrerer Elemente ermöglicht.
Mit dieser Klasse wird die Liste der Tracks realisiert.

Button
Stellt ein Schaltflächen-Steuerelement von Windows dar.

Tooltip
Stellt ein kleines rechteckiges Popupfenster dar.
Wird beim Doppelklick auf eine Datei, als Anzeige für die Übertragungsfehler verwendet.

Label
Stellt eine Standardbezeichnung von Windows dar.

5.4.3 Form1_Main.vb

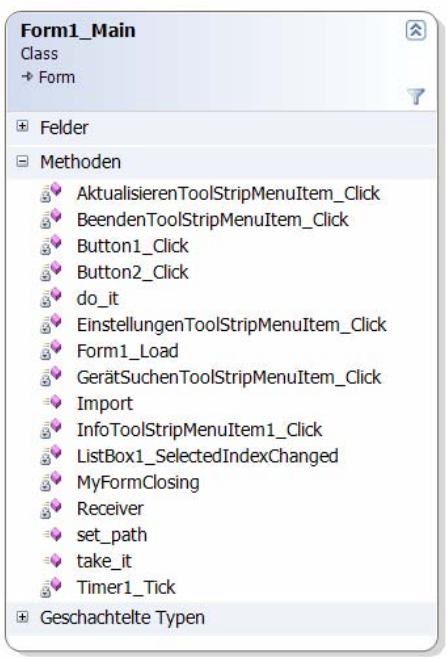


Bild 15

In Bild 15 werden die verwendeten Methoden vom Hauptprogramm dargestellt.
Wann, welche Methode ausgeführt wird ist teilweise selbsterklärend.
Die Methoden, welche mit einem _Click enden, werden durch einen Mausklick ausgeführt. Z.B. wenn der Button1 angeklickt wird, dann wird die Methode Button1_Click ausgeführt.
Timer1_Tick wird, wenn der Timer gestartet ist, immer nach einem eingestellten Zeitintervall ausgeführt.
Listbox1_SelektedIndexChanged wird abgearbeitet, wenn ein Listeneintrag markiert wird.
Form1_Load und MyFormClosing werden beim Starten und beim Beenden des Programms ausgeführt.
Die restlichen Methoden werden von anderen Methoden aufgerufen.
Auf den folgenden Seiten wird genauer auf die einzelnen Methoden eingegangen.

5.4.3.1 ToolStripMenüItem_Click

Bild 16

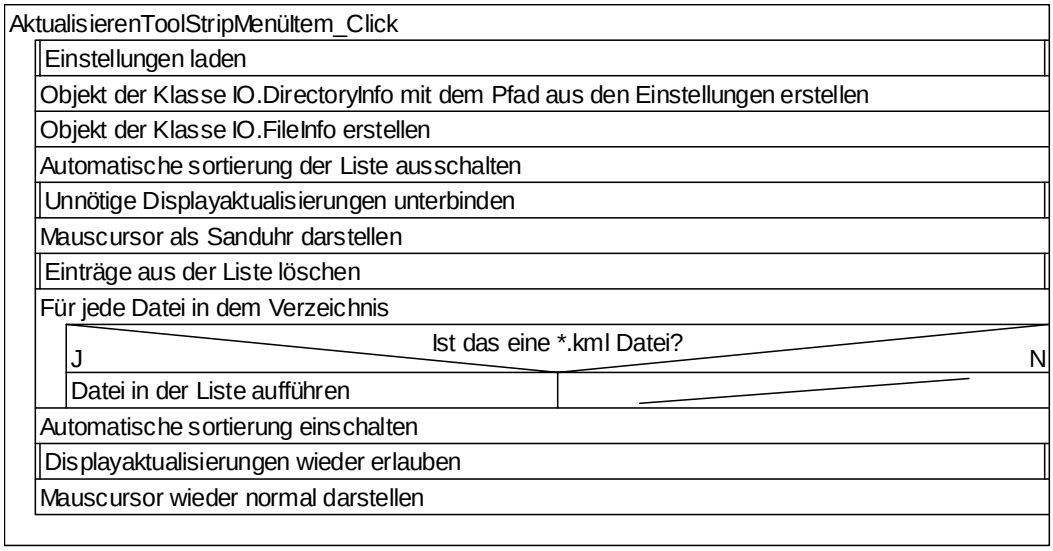
ToolStripMenuItems sind die Einträge in der Menüleiste.

Datei -> Beenden

Methode: Private Sub BeendenToolStripMenuItem_Click
Ereignis: Klicken auf das Menüitem Datei -> Beenden
Beschreibung: Beendet das Programm. Löst beim Beenden das Ereignis MyFormClosing aus.

Bearbeiten -> Aktualisieren

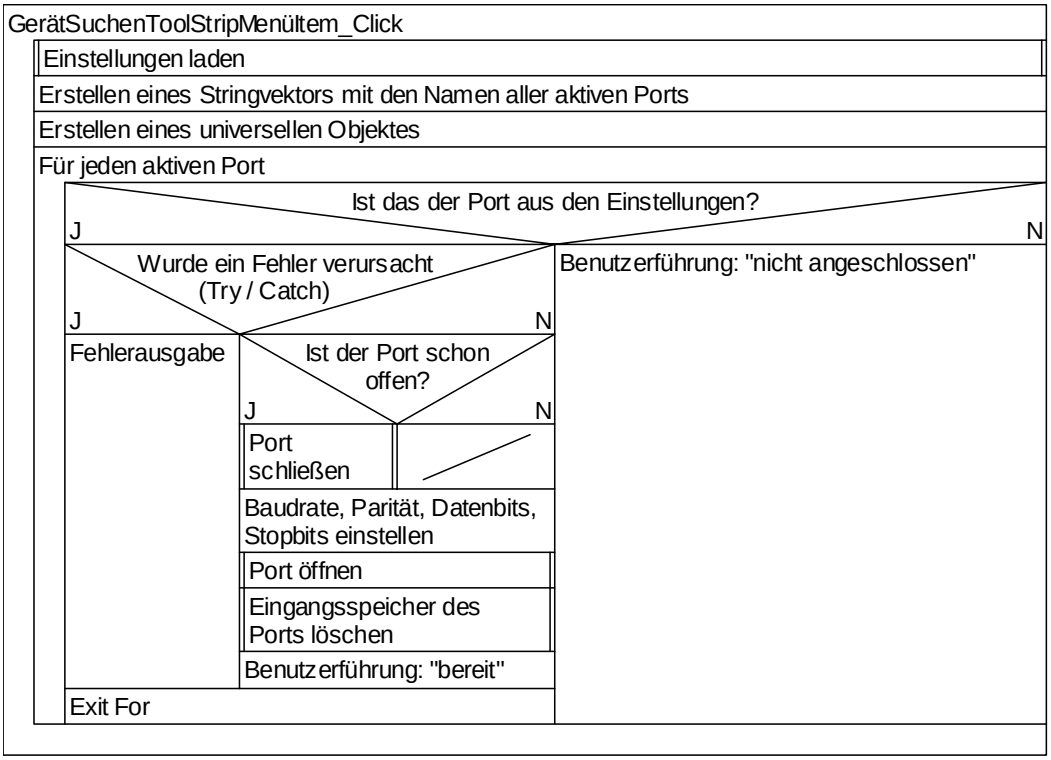
Methode: Private Sub AktualisierenToolStripMenuItem_Click
Ereignis: Klicken auf das Menüitem Bearbeiten -> Aktualisieren
Beschreibung: Aktualisiert die Liste mit den Dateieinträgen.


AktualisierenToolStripMenüItem_Click
Einstellungen laden
Objekt der Klasse IO.DirectoryInfo mit dem Pfad aus den Einstellungen erstellen
Objekt der Klasse IO.FileInfo erstellen
Automatische sortierung der Liste ausschalten
Unnötige Displayaktualisierungen unterbinden
Mauscursor als Sanduhr darstellen
Einträge aus der Liste löschen
Für jede Datei in dem Verzeichnis
Ist das eine *.kml Datei?
J
N
Datei in der Liste aufführen
Automatische sortierung einschalten
Displayaktualisierungen wieder erlauben
Mauscursor wieder normal darstellen

Struktogramm 21

Bearbeiten -> Gerät suchen

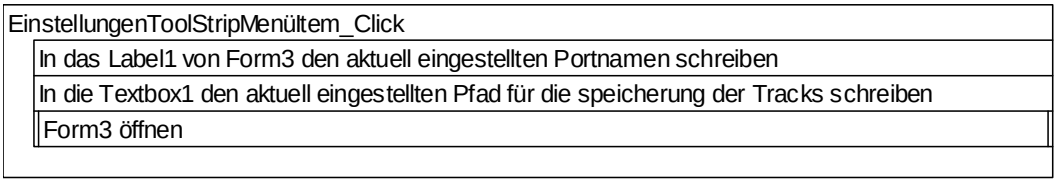
Methode: Private Sub GerätSuchenToolStripMenuItem_Click
Ereignis: Klicken auf das Menüitem Bearbeiten -> Gerät suchen
Beschreibung: Schaut ob ein Gerät mit der eingestellen COM Nummer an den Computer angeschlossen ist.



Struktogramm 22

Bearbeiten -> Einstellungen

Methode: Private Sub EinstellungenToolStripMenuItem_Click
Ereignis: Klicken auf das Menüitem Bearbeiten -> Einstellungen
Beschreibung: Öffnet das Fenster für die Einstellungen und übergibt die aktuellen Einstellungen.


EinstellungenToolStripMenültem_Click
In das Label1 von Form3 den aktuell eingestellten Portnamen schreiben
In die Textbox1 den aktuell eingestellten Pfad für die speicherung der Tracks schreiben
Form3 öffnen

Struktogramm 23

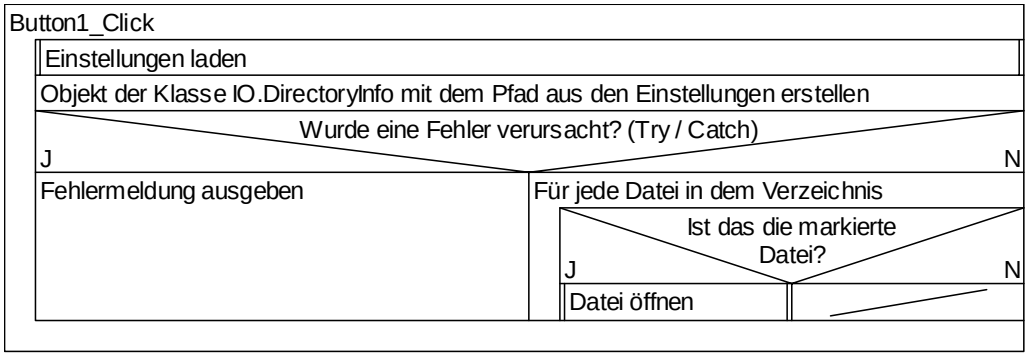
Info -> Info

Methode: Private Sub InfoToolStripMenuItem_Click
Ereignis: Klicken auf das Menüitem Info -> Info
Beschreibung: Öffnet das Fenster mit einer Information über das Programm.

5.4.3.2 Button_Click

Button1

Methode: Private Sub Button1_Click
Ereignis: Klicken auf den Button1
Beschreibung: Öffnet Google Earth mit der markierten Datei

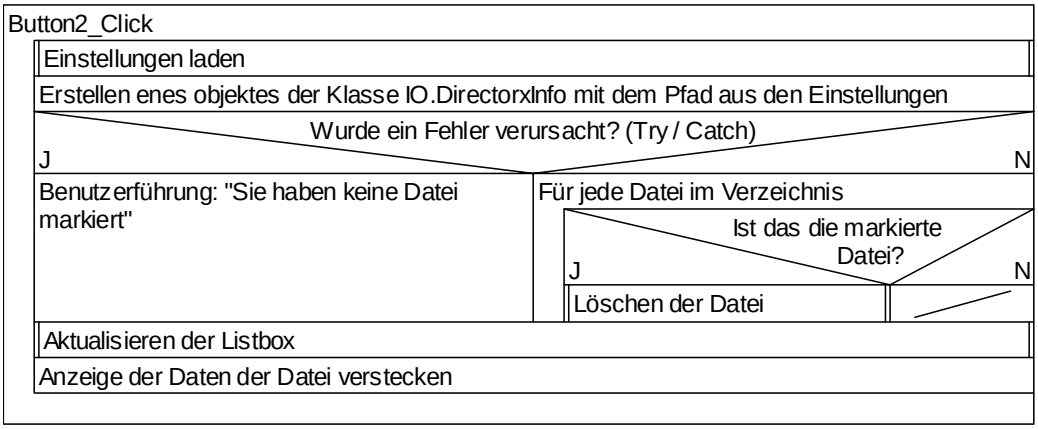


Struktogramm 24

Button2

Methode: Private Sub Button2_Click
Ereignis: Klicken auf den Button2
Beschreibung: Löscht die markierte Datei

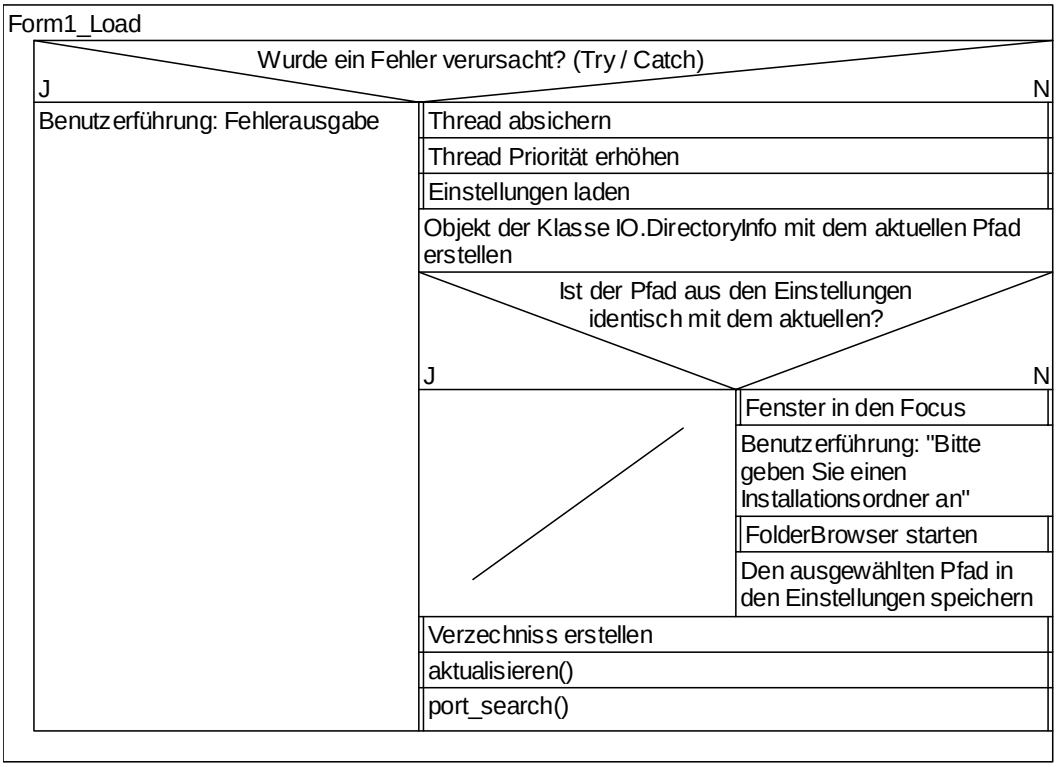
Struktogramm 25



5.4.3.3 Load and Close

Load

Methode: Private Sub Form1_Load
Ereignis: Starten der Software
Beschreibung: Konfiguriert die Software

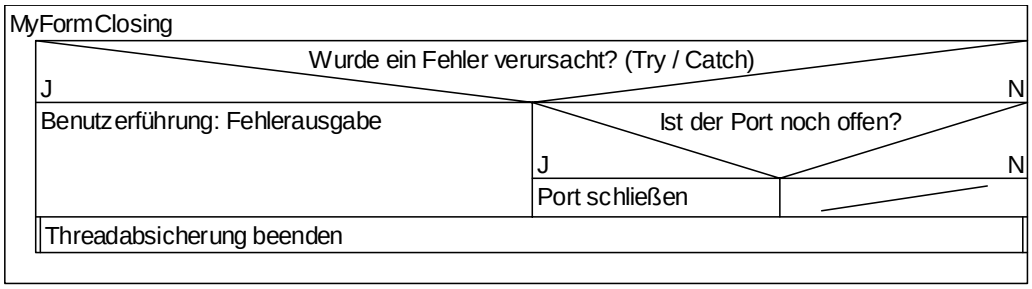


Struktogramm 26

Close

Methode: Private Sub MyFormClosing
Ereignis: Beenden der Software
Beschreibung: Schließt gegebenenfalls den Port und beendet die Thread-absicherung.

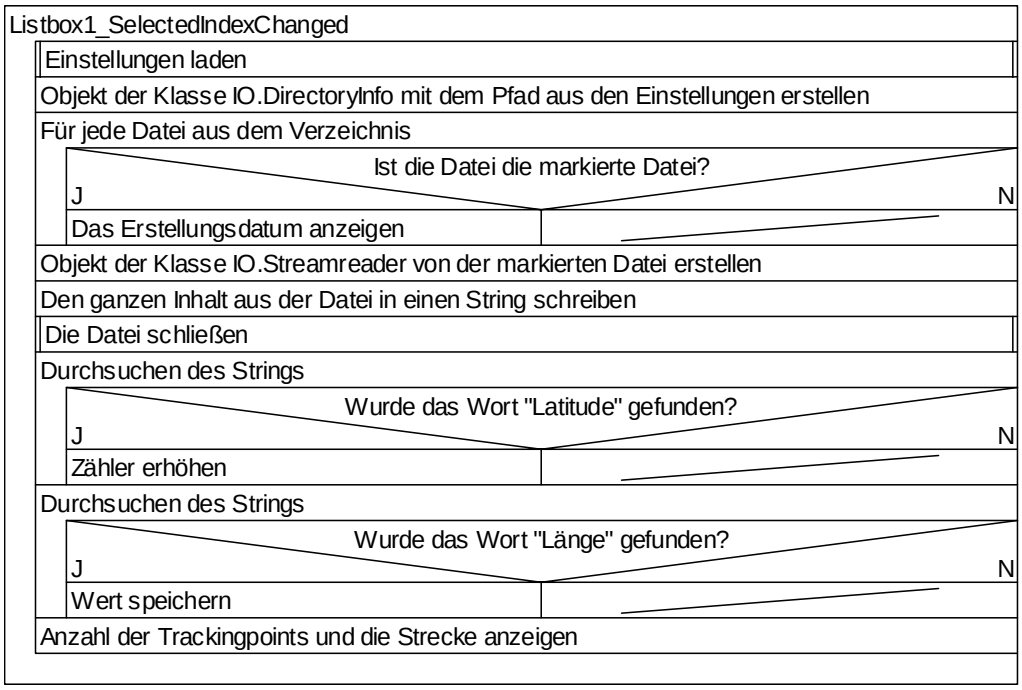
Struktogramm 27



5.4.3.4 SelectedItemChanged

Listbox1

Methode: Private Sub Listbox1_SelectedItemChanged
Ereignis: Markieren einer Datei aus der Liste
Beschreibung: Zeigt die Informationen über den Track an



Struktogramm 28

5.4.3.5 Timer_Tick

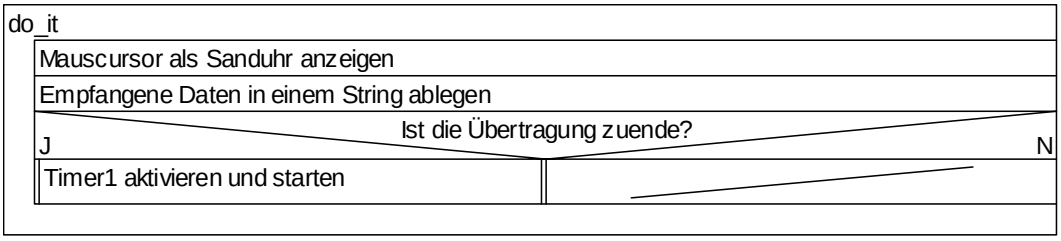
Timer1

Methode: Private Sub Timer1_Tick
Ereignis: Abgelaufenes Zeitintervall des Timer1
Beschreibung: Portal zwischen den Threads. Dient nur dem Aufruf der Methode Import().

5.4.3.6 DataReceived

Methode: Private Sub Receiver
Ereignis: Daten werden über die USB Schnittstelle empfangen
Beschreibung: Nimmt die Daten entgegen und übergibt sie dem Mainthread

Methode: Private Sub do_it
Ereignis: Daten werden über die USB Schnittstelle empfangen
Beschreibung: Nimmt die Daten entgegen und übergibt sie dem Mainthread



Struktogramm 29

5.4.4 Form2_Info.vb

Diese Datei beinhaltet einzig und alleine ein Fenster mit einer Information über die Software.
Das Fenster wird über das Hauptprogramm geöffnet. Die einzige Funktion verbirgt sich hinter dem Button, welcher das Fenster wieder schließt.

Bild 17

5.4.5 Form2_Einstellungen.vb

Diese Datei beinhaltet das Fenster, in dem die Einstellungen angezeigt und verändert werden können.
Es wird über das Hauptprogramm geöffnet. Bevor das Fenster jedoch angezeigt wird, werden die Daten der aktuell eingestellten Schnittstelle und der Speicherpfad übergeben sowie die verfügbaren Ports ermittelt.

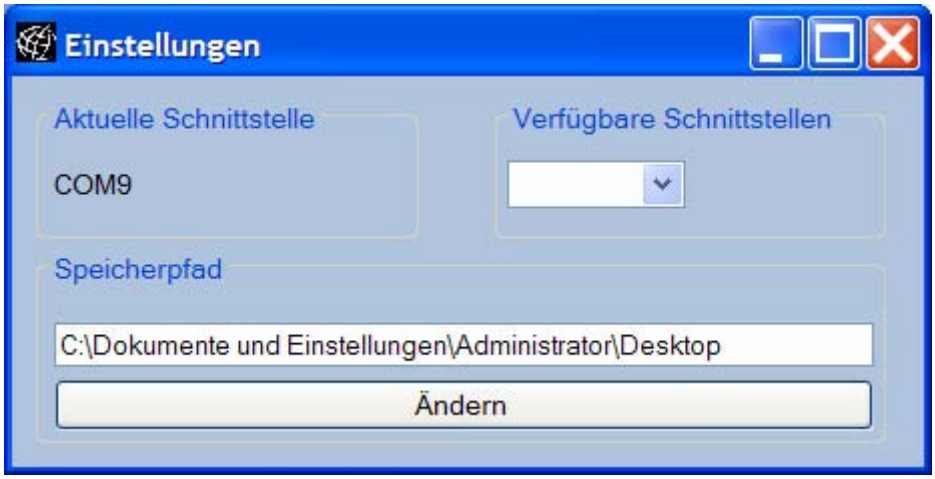


Bild 18

Wird eine verfügbare Schnittstelle ausgewählt, wird eine Funktion des Hauptprogrammes aufgerufen, welche den neuen Port öffnet. Zudem wird die neue Einstellung abgespeichert, so dass diese beim nächsten Start des Programmes wieder aktiv ist. Anschließend wird das Fenster geschlossen.

Wird der Button „Ändern“ betätigt, wird wiederum eine Funktion des Hauptprogrammes aufgerufen, welche das Fenster (FolderBrowserDialog), zum Auswählen eines Speicherpfades, öffnet und den ausgewählten Speicherpfad abspeichert.
Anschließend wird der neue Speicherpfad in der Textbox angezeigt.

Kapitel 6
Bedienungsanleitung

- **Handgerät**
- **PC Software**
- **Kontrollelemente**
- **Mögliche Fehler**

=> Wie wird der GPS-Tracker richtig bedient?

6.1 Handgerät

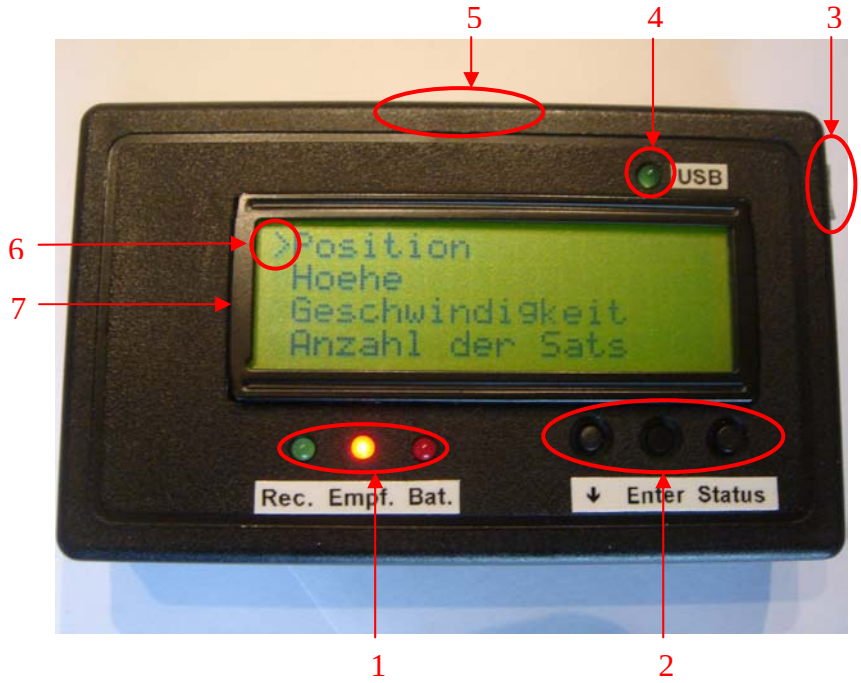


Bild 19

6.1.1 Legende

1. Status LEDs
   - Rec.: leuchtet bei einer Aufnahme
   - Empf.: leuchtet bei Empfang
   - Bat.: leuchtet bei schwachem Batteriestand

2. Taster
   - ↓: zum Blättern und um aus einem Menüpunkt wieder zurückzukehren
   - Enter: zum bestätigen eines Menüpunktes
   - Status: zum Aufrufen des Statusmenüs (Taste gedrückt halten)

3. USB Buchse: für die Anbindung an einen Computer

4. USB Status: leuchtet wenn das Gerät an einen Computer angeschlossen ist

5. Einschalter: zum Ein- und Ausschalten des Gerätes

6. LC-Display: zur Menüführung und Informationsausgabe

7. Markierung: markiert den angewählten Menüpunkt

6.1.2 Menüaufbau

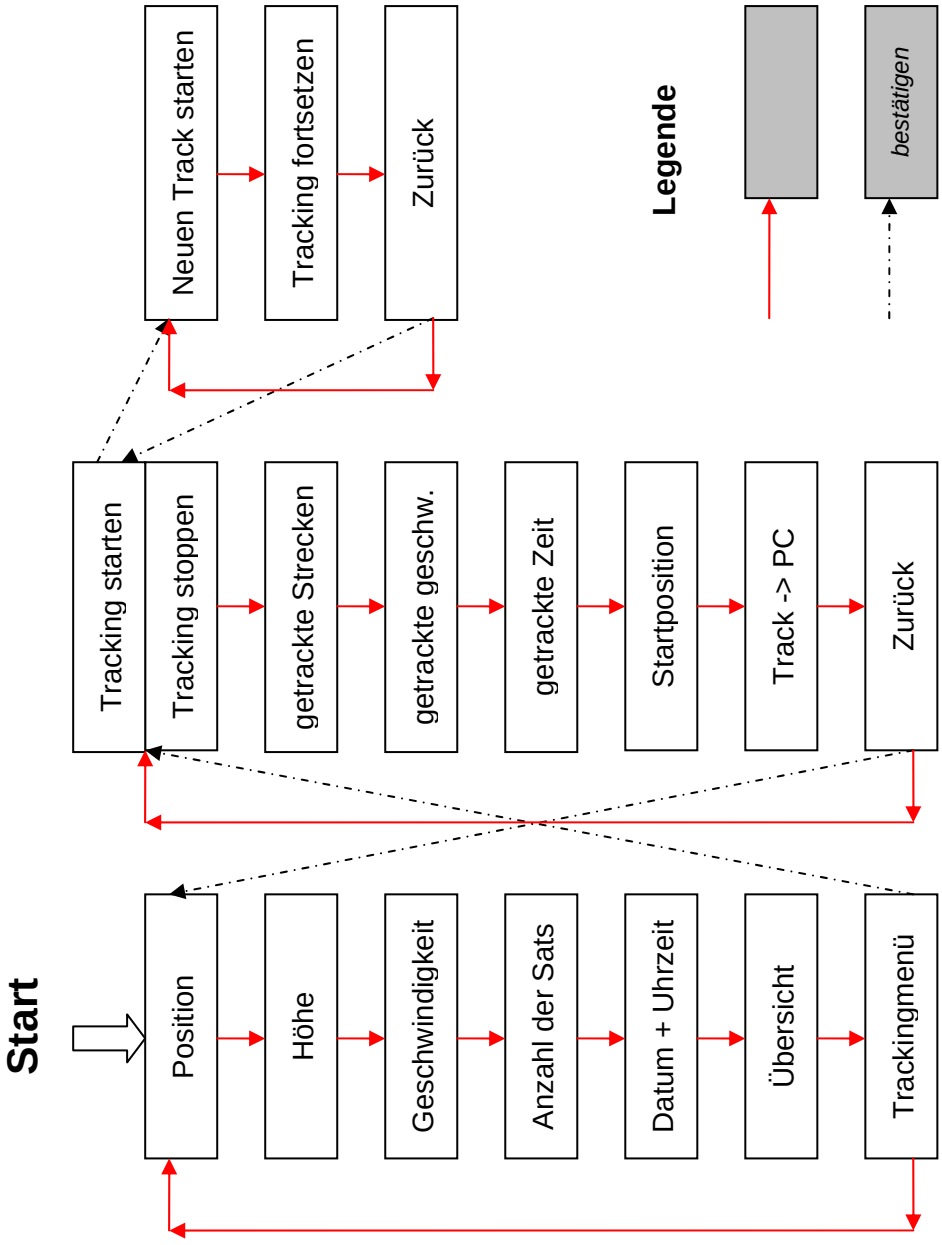


Bild 20

6.1.3 Bedienung

Allgemeines

Das Menü wurde so gestaltet, das es sehr intuitiv bedienbar ist.
Das Gerät kann nicht in dem Sinne falsch bedient werden, dass es abstürzt oder gar zerstört wird.
Um einen Track aufzunehmen muss Empfang bestehen. Die besten Vorraussetzungen dafür sind im Freien mit guter Sicht zum Himmel. Je nach dem ob ein Warmstart oder ein Kaltstart durchgeführt wird, kann es bis zu 15 min dauern bis eine gute Verbindung zu den Satelliten aufgebaut werden kann.

Es bestehen zwei Möglichkeiten einen Track aufzuzeichnen:

1. Einen neuen Track starten. Hier werden die Daten der letzten Aufnahme gelöscht und ein neuer Track gestartet.
2. Den letzten Track fortsetzen. Hier wird der im Speicher vorhandene Track fortgesetzt.

Wenn eine Aufzeichnung durch das Ausschalten des Gerätes beendet wird, gehen die Daten verloren. Der Track muss im Trackingmenü über den Menüpunkt „Tracking stoppen“ beendet werden. Anschließend kann das Gerät ohne Datenverlust ausgeschaltet werden
Um einen aufgenommenen Track in „Google Earth“ darzustellen, muss das Gerät, über das mitgelieferte USB Kabel an den Computer angeschlossen werden.

Track aufnehmen

1. Gerät einschalten
2. Warten bis Empfang besteht (die mittlere, rote LED erlischt)
3. Mit der Taste „↓“ solange blättern, bis der Menüpunkt „Trackingmenü“ mit dem „>“ Zeichen markiert ist
4. Diesen Menüpunkt mit der Taste „Enter“ bestätigen
5. In dem Trackingmenü muss jetzt der Menüpunkt „Tracking starten“ markiert sein
6. Diesen Menüpunkt mit der Taste „Enter“ bestätigen
7. Mit der Taste „↓“ eine Aufnahmeoption auswählen („Neuen Track starten“ oder „Track fortsetzen“ markieren)
8. Diesen Menüpunkt mit der Taste „Enter“ bestätigen
9. Die grüne LED leuchtet auf und auf dem Display ist der Menüpunkt „Tracking stoppen“ markiert – es wird ein Track aufgezeichnet
10. Zum Beenden einer Aufnahme muss der Menüpunkt „Tracking stoppen“ markiert und mit „Enter“ bestätigt werden

6.1.4 Akku laden

Um den Akku zu laden, muss das Gerät geöffnet werden. Dazu sind beide Schrauben an der Unterseite des Gerätes zu lösen. Sind die Schrauben entfernt, kann das Gerät vorsichtig aufgeklappt werden.

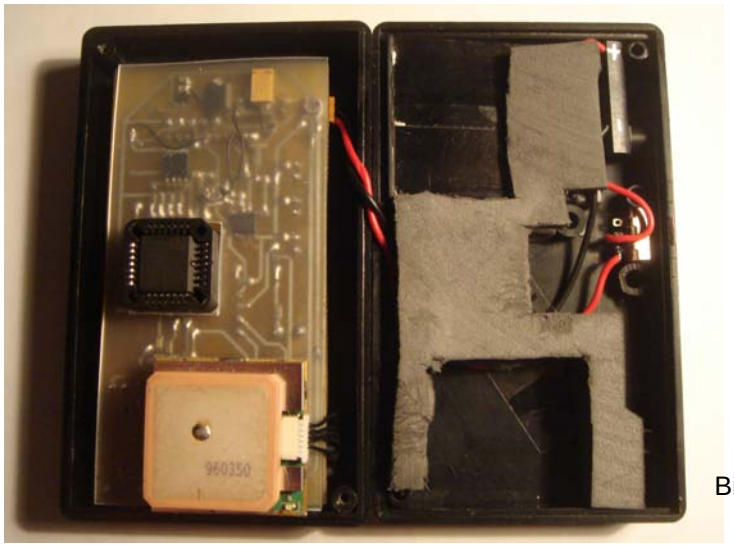


Bild 21

In dem Gehäuse befindet sich zur Fixierung der einzelnen Elemente Moosgummi. Bevor der Akku entnommen werden kann, muss der Moosgummi noch entfernt werden. Der Akku kann dann einfach herausgezogen werden.
Dieser ist ausschließlich in der mitgelieferten Ladestation zu laden.

6.2 PC Software

6.2.1 Erstinbetriebnahme

Wird das Gerät das erste Mal an den Computer angeschlossen, müssen zuerst zwei Treiber installiert werden. Diese werden automatisch installiert.
Der erste ist für den USB Controller des Handgerätes, der zweite ist für den virtuellen Comport.
Nach Installation der Treiber müssen nun ein paar Einstellungen in der Systemsteuerung vorgenommen werden. In dem Geräte-Manager (Systemsteuerung - System - Reiter „Hardware“ – Geräte-Manager) ist jetzt unter der Kategorie Anschlüsse (COM und LPT) der Menüeintrag USB Serial Port (COM X) erschienen.
Durch einen Doppelklick auf diesen Menüeintrag erscheint ein Fenster mit den Eigenschaften des USB Serial Ports. Dort müssen unter dem Reiter „Port Settings“ die Einstellungen wie auf Bild 22 zu sehen, Vorgenommen werden.

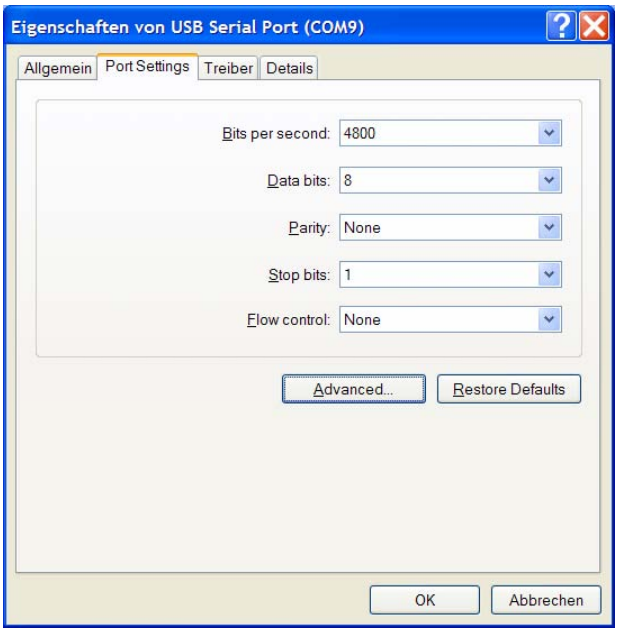


Bild 22

Über den Button „Advanced…“ wird ein neues Fenster geöffnet. Hie müssen die Einstellungen wie auf Bild 23 zu sehen, übernommen werden.
Die COM Port Nummer kann frei gewählt werden, aber muss später auch in der Software eingestellt werden.

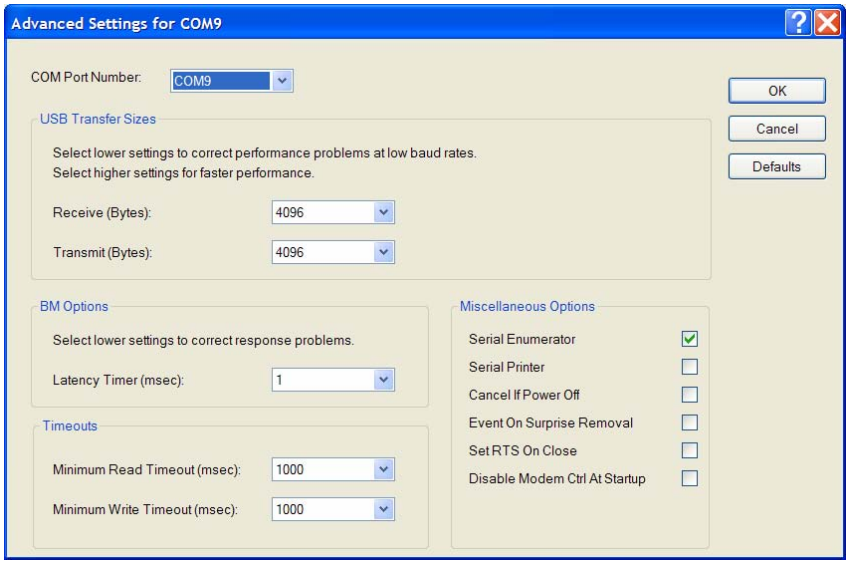


Bild 23

Wurden diese Schritte erfolgreich durchgeführt kann die GPS-Tracker Software, durch einen Doppelklick auf die Datei GPS-Tracker.exe, gestartet werden.
Wenn diese zum ersten Mal unter einem neuen Pfad gestartet wird, wird nach dem gewünschten Verzeichnis gefragt, in dem die Tracks später gespeichert werden sollen. Dieser kann aber nachträglich auch wieder geändert werden.
Jetzt muss noch die COM Port Nummer eingestellt werden. Diese kann unter dem Menü „Bearbeiten – Einstellungen“, wie in Bild 24 zu sehen, über die Auswahl in „Verfügbare Schnittstellen“ eingestellt werden.

Bild 24

Das Fenster schließt sich dann automatisch und im Hauptfenster steht unter Serial port „bereit“.
Ist das der Fall, kann zu jeder Zeit die Übertragung von dem Handgerät aus gestartet werden.

6.2.2 Bedienung

Nachdem die Einstellungen der Erstinbetriebnahme durchgeführt wurden, ist die Software funktionsbereit.

Track übertragen (1)

1. Handgerät über das USB Kabel an den Computer anschließen
2. Software starten (Doppelklick auf GPS-Tracker.exe)
3. Menüpunkt „Track -> PC“ im Trackingmenü des Handgerätes auswählen
4. Warten bis der SaveFileDialog erscheint
5. Speicherort auswählen und gewünschten Dateinamen eingeben
6. Datei in der Liste markieren und den Button „Öffnen“ betätigen
7. „Google Earth“ wird gestartet und zoomt automatisch auf den Track

Track übertragen (2)

1. Software ist gestartet und Gerät noch nicht angeschlossen
2. Gerät anschließen
3. Bearbeiten – Gerät suchen
4. Weiter wie oben ab Punkt 3

COM Port Nummer einstellen (ändern)

1. Gerät anschließen
2. Software starten
3. Bearbeiten -> Einstellungen auswählen
4. Unter Verfügbare Schnittstellen die Port Nummer auswählen

Verzeichnis zur Speicherung ändern / Verzeichnis der Listeneinträge andern

1. Software starten
2. Bearbeiten -> Einstellungen
3. In dem Feld „Speicherpfad“ den Button „ändern“ betätigen
4. Gewünschtes Verzeichnis auswählen oder erstellen

6.3 Kontrollelemente

1. Bei angeschlossenem Gerät muss die USB LED beim Handgerät leuchten
2. Bei angeschlossenem Gerät muss in dem Feld „Serial port“ „bereit“ stehen
3. Bei Doppelklick auf einen Track in der Liste, erscheint die Fehleranzahl von der Übertragung

6.4 Mögliche Probleme

Problem: USB LED leuchtet bei angeschlossenem Gerät nicht.
Ursache: Treiber ist noch nicht oder fehlerhaft installiert.
Behebung: Das Gerät erneut einstecken und den Treiber automatisch installieren lassen. Falls das nicht funktioniert, das Gerät ausstecken, den Treiber mit der mitgelieferten Software (FT Clean) entfernen und das Gerät erneut einstecken.

Problem: In dem Feld „Serial port“ wird nicht „bereit“ angezeigt.
Ursache(1): Das Gerät wurde nach dem Start der Software eingesteckt.
Behebung: Unter „Bearbeiten“ „Gerät suchen“ auswählen.
Ursache(2): Die Schnittstelle wurde nicht, oder fehlerhaft eingestellt.
Behebung: Die Erstinbetriebnahme, ohne Neuinstallation der Treiber, wiederholen.

Problem: Das Gerät zeigt keine Reaktion nach dem Einschalten.
Ursache: Der Akku ist leer.
Behebung: Akku, mit dem mitgelieferten Ladegerät, wie in Abschnitt 6.1.4 beschrieben, laden.

Kapitel 7
Fazit

=> Wozu das alles?

Als ich mit der Technikerarbeit anfing, wusste ich noch nicht so genau, auf was ich mich eingelassen hatte. Ich war mir im Klaren, dass ich mir nicht gerade eine einfache Arbeit herausgesucht hatte. Mir ging es jedoch hauptsächlich darum, dass es eine Arbeit ist, welche mich interessiert und mich auch herausfordert.
Mir sind mehrere Fehler unterlaufen. In der Planung sowie in der Entwicklung, doch ich wollte keinen davon, nicht begangen haben. Aus jedem Fehler der mir unterlief, habe ich eine Menge Fachwissen und aber auch Wissen über mich selbst, erlangt.
Im Nachhinein könnte ich in vieler Hinsicht sagen wie ich es hätte besser machen können, doch dazu waren eben genau diese Fehler notwendig.
Die Übertragung zum PC würde ich jetzt mit einer Sende- und einer Empfangsleitung realisieren.
Die vertikale Geschwindigkeit würde ich, wenn überhaupt, mit einem umfangreicheren Algorithmus, der auf komplexeren Berechnungen beruht, verwirklichen.
Der GPS-Tracker ist nicht perfekt. Es gibt mehrere Details, die anders eventuell besser gelöst wären. Doch dies ist das erste große Projekt, das ich bis ins Detail selbst geplant und entwickelt habe. Die Erfahrungen, welche ich aus dieser Arbeit gewonnen habe, sind wichtiger als ein perfektes Produkt.
Ich habe mir für die Realisierung dieses Projekts die Programmiersprache „Visual Basic 2005“ selbst angeeignet. Schon alleine dieser Aspekt war die Arbeit wert. Im Laufe der Entwicklung haben sich meine Programmierkenntnisse in der Sprache „C“ immens gesteigert. Zudem habe ich einige Erfahrungen in der Entwicklung, der Fehlersuche und im Aufbau von elektronischen Schaltungen erlangt. All dies wird mir in meiner Zukunft vieles erleichtern und eventuell auch vieles ermöglichen.
Abschließend ist zu sagen, dass ich froh bin, mir diese Herausforderung gestellt zu haben. Ich verbrachte sehr viel Zeit, teilweise auch schlaflose Nächte, damit dieses Projekt zu verwirklichen. Doch stände ich wieder am Anfang der Techniker-ausbildung, würde ich mich wieder dazu entscheiden.

Kapitel 8
Anhang

- **Eidesstattliche Erklärung**
- **Glossar**
- **Quellen**
- **Inhalt der beigefügten CD**

8.1 Eidesstattliche Erklärung

Hiermit versichere ich, Björn Hackel, dass die vorliegende Technikerarbeit von mir selbständig geplant, erarbeitet und angefertigt wurde.
Alle Quellen welche der Informationsbeschaffung dienten, sind aufgelistet.

_____ _____

Ort, Datum Unterschrift

8.2 Glossar

ASCII	American Standard Code for Information Interchange ist eine 7-Bit-Zeichenkodierung und bildet die Grundlage für spätere mehrbittige Zeichensätze und -kodierungen. Die Zeichenkodierung definiert 128 Zeichen, davon 33 nicht-druckbare sowie folgende 95 druckbare.
EEPROM	EEPROM (Electrically Erasable Programmable Read-Only Memory, wörtlich: elektrisch löschbarer, programmierbarer Nur-Lese-Speicher) ist ein nichtflüchtiger, elektronischer Speicherbaustein.
EMV	Elektromagnetische Verträglichkeit
Flag	Binäre Variable, kennzeichnet einen Zustand
Google Earth	Softwareprogramm für einen PC. Es stellt einen virtuellen Globus mit Koordinaten dar.
GPS	Global Positioning System. Sattelitengestütztes Positionsbestimmungs- und Ortungssystem
I²C Bus	Inter-Integrated Circuit. Iist ein serieller Datenbus
Interrupt	Programmunterbrechung, bedingt durch ein Ereignis. Als Folge wird eine Interrupt-Service-Routine abgearbeitet.
Hyper Terminal	Hyper Terminal ist ein Kommunikationsprogramm
ISR	Interrupt-Service-Routine.
Klasse	Ist in der Objektorientierung ein abstrakter Oberbegriff für die Beschreibung der gemeinsamen Struktur und des gemeinsamen Verhaltens von Objekten (Klassifizierung).
*.kml Datei	Format einer Datei, welche in Google Earth eingebunden werden kann
LCD	Liquid Crystal Display. Ein Flüssigkristallbildschirm oder eine Flüssigkristallanzeige, ist ein Bildschirm oder eine Anzeige in dem die Eigenschaft von Flüssigkristallen genutzt wird, die

	Polarisationsrichtung von Licht zu beeinflussen, um Zeichen, Symbole und Bilder anzuzeigen.
LED	Light emitting Diode. Auch als Leuchtdioden bekannt. Eine energiesparende Lichtquelle.
LPC	Mikrokontroller Familie der Firma NXP
Objekt	Ein Objekt bezeichnet in der Objektorientierung ein Exemplar eines bestimmten Datentypes oder einer bestimmten Klasse.
SD Karte	Eine SD Memory Card (Kurzform für Secure Digital Memory Card; dt. sichere digitale Speicherkarte) ist ein digitales Speichermedium. Kann eine Kapazität von bis zu 32 GB haben.
SMD	Der Begriff oberflächenmontierbares Bauteil (englisch surface-mounted device, SMD) ist ein Fachbegriff aus der Elektronik. Diese Bauelemente haben im Gegensatz zur Durchsteckmontage keine Drahtanschlüsse, sondern werden mittels lötfähiger Anschlussflächen direkt auf eine Leiterplatte gelötet.
Track	Strecke, die das GPS Gerät aufgezeichnet hat.
UART	UART ist die Abkürzung für Universal Asynchronous Receiver Transmitter. Die Funktion ist, einen seriellen digitalen Datenstrom mit einem fixen Rahmen aufzubauen, welcher aus einem Start-Bit, fünf bis maximal neun Datenbits, einem optionalen Parity-Bit zur Erkennung von Übertragungsfehlern und einem Stopp-Bit besteht. Das UART dient sowohl zum Senden als auch zum Empfangen von Daten.
USB	Der Universal Serial Bus ist ein serielles Bussystem zur Verbindung eines Computers mit externen Geräten.
µC	Mikrocontroller

8.3 Quellen

Websites

http://thomaspfeifer.net/
http://www.kowoma.de/
http://www.innovatic.dk/knowledg/SerialCOM/SerialCOM.htm
http://www.galileocomputing.de/openbook/visual_basic/
http://www.mikrocontroller.net/
http://msdn.microsoft.com/de-de/library/
http://de.wikipedia.org/
http://earth.google.de/kml/whatiskml.html

Bücher

Visual Basic 2005
C für PCs
Keil C51 / Philips LPC900, Hardware – Software –Toolchain

8.4 Inhalt der beigefügten CD

- Quellcode Mikrocontroller
- Quellcode PC
- Dokumentation als PDF-Datei
- Datenblätter